

**kbeeworkflow**

Concepts and Architecture Manual.



# kbee.workflow

## Concepts and architecture

### Document Properties

Projecto:

**kbee.workflow**

Document Title:

**Concepts and architecture**

Version date:

**12/04/07**

Version number:

**V005**

Author:

**Alejo Ferraria, Alejandro Tolomei**



## Table of Contents

Table of Contents.....	3
<b>1. Introduction .....</b>	<b>5</b>
1.1 Product strategy and features .....	7
<b>2. kbee.workflow components .....</b>	<b>8</b>
2.1 kbee process designer .....	8
2.2 kbee.workflow server .....	8
2.3 kbee.OLAP Server.....	9
2.4 kbee WQL.....	9
2.5 kbee WQL monitor.....	9
<b>3. kbee.workflow Architecture.....</b>	<b>10</b>
3.1 Client application tier.....	11
3.2 Client servant tier .....	11
3.3 Workflow server tier .....	11
3.4 Connectors tier.....	11
<b>4. Process Management: Definition .....</b>	<b>12</b>
4.1 Cases .....	12
4.2 Procedures.....	12
4.3 Task.....	12
4.4 Work Items.....	12
<b>5. Petri nets for process definition .....</b>	<b>13</b>
5.1 Classic Petri nets .....	13
5.2 High-level Petri nets.....	14
5.3 Process Definition with Petri nets.....	15
5.4 Routing Constructs .....	16
5.5 Triggering.....	19
<b>6. kbee Workflow Query Language (kbee.WQL).....</b>	<b>20</b>

6.1 Native and Extended Models .....	20
6.2 kbee WQL language.....	21
<b>7. Analytical server and OLAP reports .....</b>	<b>23</b>
<b>Appendix I.....</b>	<b>24</b>
kbee.workflow license.....	24





# 1. Introduction

## **Process Management in organizations**

Today, business processes within an organization require a number of steps in order to solve a problem or produce a deliverable. These steps may imply the execution of individual tasks (members of the organization or even external users) together with the assistance of tools with a different level of automation, or steps that can be directly and completely executed by applications or systems. Examples of such processes are: delivering an order, solving a complaint or the management of a file throughout its lifecycle.

We can define the term process as a specific sequence of steps used to solve a particular problem. Generally, the decision of which steps are to be followed, as well as who and when should execute them is determined by the expertise or knowledge of the members of the organization involved in said process.

Moreover, these members usually hand over the process to the person responsible for executing each step. For instance: a file or an order that is taken to various offices by a person in a folder.

## **The growing complexity of process**

While the complexity and number of processes grow, managers need to control and monitor the flow of work within the organization by being able to track at any time the number and state of processes under execution.

Due to the ever increasing complexity of work, it is critical to ensure that process management follows the steps defined by the company's managers (compliance with quality standards such as ISO is a clear example of it).

Manual management and monitoring become cumbersome as the complexity and number of processes grows. Also, modifying the company's IT applications can become a difficult and expensive task when work processes change.

Moreover, the globalization and decentralization of companies, together with the increasing trend of customers and partners' incorporation to work dynamics arises the necessity of creating a robust and versatile layer for process execution.

## **Business Process Management Software (BPM)**

Business process management is infrastructure software designed to automate the execution and control of processes. Based on a graphical process model, it determines who (or which application) should execute a task, as well as at what time it should be executed to solve a problem. It also encompasses monitoring tools that provide indicators of quantity, history and state of processes.

BPM software should be independent from the applications carrying out the tasks; so that routing conditions and assignment rules can be easily modified without requiring changes to the existing applications.



### **kbee.workflow**

kbee.workflow is simple and versatile BPM software. It encompasses the essential tools to create a sophisticated and flexible process layer in a business application. It is a solid platform that has been proven in applications with thousands of users and processes under execution.

kbee.workflow is a Java technology-based platform which has been built on Open Standards. It provides components for process definition integrated to the Eclipse framework, tools to monitor the processes under execution, a multidimensional OLAP server for analytical queries, together with an OQL engine (Object Query Language) which simplifies the integration of work consoles with other applications.



## 1.1 Product strategy and features

### Simple and focused

The product has been designed as a powerful and simple tool for developers. The aim is to provide developers with the essential elements to build workflow applications, together with the infrastructure to extend the functionality for different domains through the use of libraries.

Its key components are: a powerful and simple graphical specification language offering the greatest power of expression to define processes with parallel lines of execution (based on the mathematical abstraction known as Petri Nets), an Eclipse-integrated application which enables users to define processes graphically, a simple and robust engine for the execution and control of processes, an OQL-like query language (OQL: Object Query Language) to develop indicators for monitoring progress, and an OLAP server integrated to the server for analytical reports and business intelligence.

### Fully extensible to each domain

kbee.workflow is entirely written in Java and developed as a Framework, so that its core components can be easily extended to fit new requirements. Problem-specific records or operations can be easily attached to the events fired by the workflow engine when processes change their state.

### kbee WQL. Workflow Query Language

kbee.workflow offers an OQL-like query language (Object Query Language) which enables the use of indicators to monitor the state of processes at real time.

A selection criteria can be applied both upon the native attributes of the BPM (such as the state or runtime of a process) or upon extended attributes from a specific domain (such as the number of a purchase order or a file). For instance, the developer can create indicators to monitor the number of processes in a specific state or the tasks carried out onto a specific record.

### LGPL: Entirely Open Source

kbee.workflow is backed by the Open Source community. As it is licensed under the LGPL license, it offers total freedom of use and guarantees it will not degrade in the future.

### Scalable

Its powerful and simple graphical specification language with very few primitives enables the engine to be small and efficient. Moreover, its WQL (Workflow Query Language) can scale to manage thousands of active processes.

kbee.workflow is based on Open Standards; and it has been built and integrated to such standards as JNDI, JTA, JMS, JDBC, etc.

## 2. kbee.workflow components

### 2.1 kbee process designer

Kbee process designer is an application for the graphical specification of procedures; it includes tasks and routing conditions.

It is an Eclipse platform plug-in (<http://www.eclipse.org>) that enables an integrated working environment where developers can specify procedures, Java tasks, HTML, JSP or whatever may be necessary (Chapters 4 and 5)

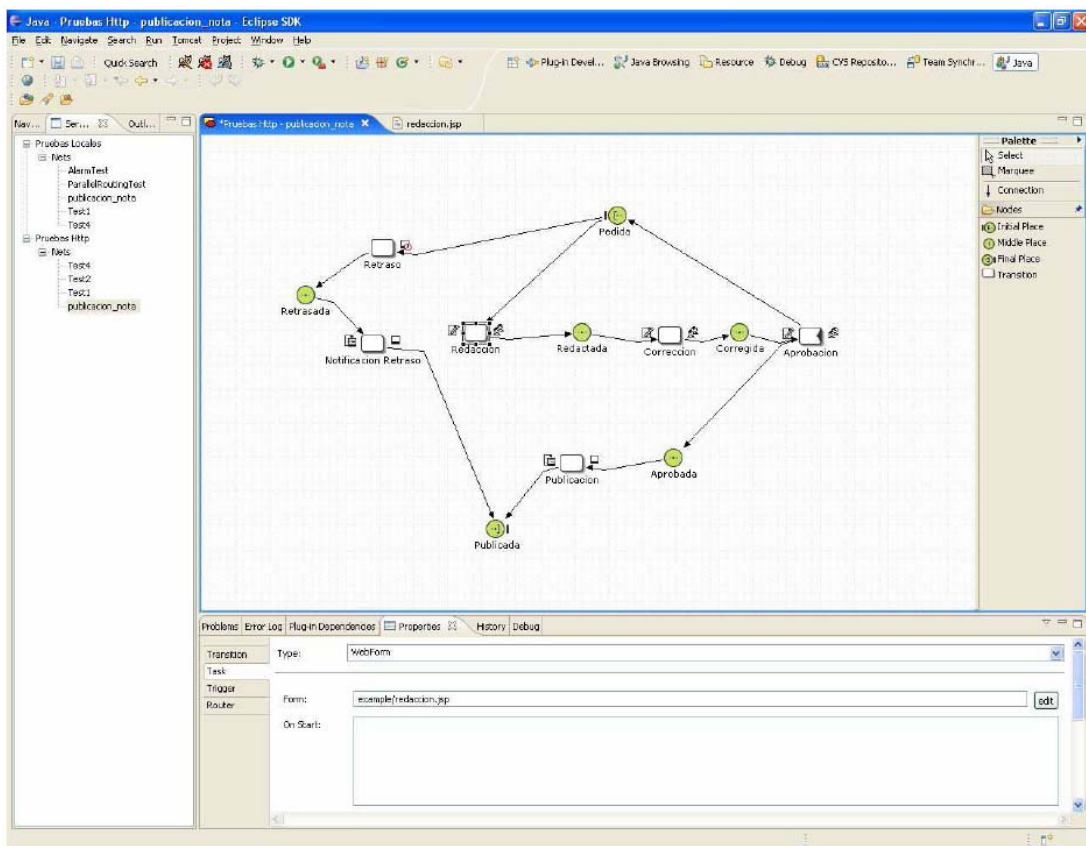


Fig. 1. kbee process designer over Eclipse

### 2.2 kbee.workflow server

Kbee workflow server is a process execution engine that executes and controls each process. It is a small and specialized server. It also has the capability to persist the changes of state in each process. Finally, it includes an event propagation mechanism to notify each change in state to the applications assigned to propagate these events to JMS queues.

#### Alarms

As a complement, the server provides a mechanism for the programming and recording of alarms. An alarm will be activated in case indicators exceed control levels and an action will be consequently executed. For instance, a signal is activated in a dashboard every time the active



processes exceed a certain number, or when a task runs longer than the expected runtime, or in the case of failed tasks in a process, etc.

### 2.3 kbee.OLAP Server

It is a multidimensional OLAP server that enables monitoring the relevant variables of each system such as: state and history of each process, tasks under execution, bottle-neck detection within processes, user-tasks reports, state of each document, etc. (Chapter 7).

### 2.4 kbee WQL

kbee WQL is a query language with a similar syntax to OQL (Object Query Language). It allows the selection and projection of native attributes or extended attributes of the workflow entities. The language enables users to query the workflow engine in analogy to queries over a Relational Database Engine (Chapter 6).

### 2.5 kbee WQL monitor

It is a query console that enables writing a WQL sentence and viewing the corresponding results.

### 3. kbee.workflow Architecture

kbee.workflow provides four-tier architecture: a client application and components tier, a client servant tier, a workflow server tier and a connectors tier.

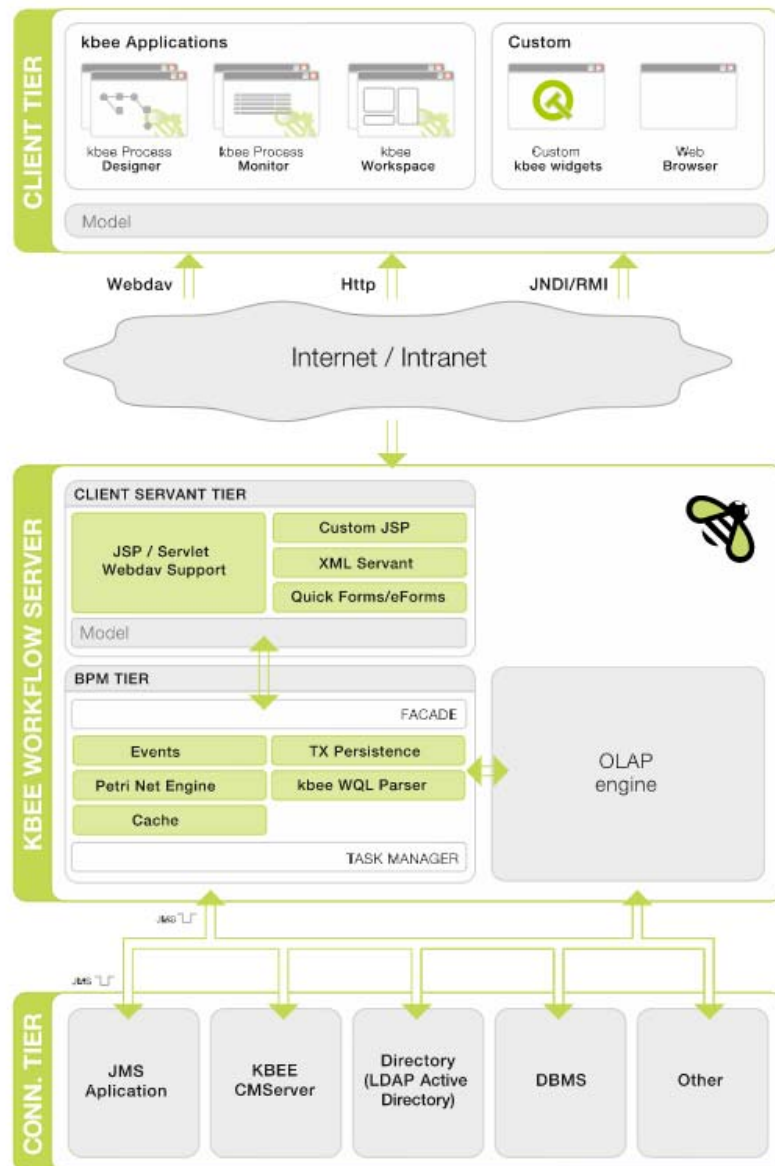


Fig. 2. kbee.workflow Architecture



### 3.1 Client application tier

The first tier is the client applications tier. This tier includes a graphical process designer, a process monitor and all the applications using any of the workflow services such as: personalized consoles, monitoring consoles or personalized dashboards.

### 3.2 Client servant tier

In this tier, a set of Java components (Web Services, Servlets, JSP pages, etc.) implement a well-defined interface of all the services provided by the workflow server so they can be accessed by HTTP.

In case the client applications work in the same server and virtual engine than the workflow applications, there is an alternative to seamlessly access to the services through the API Java defining its interface.

### 3.3 Workflow server tier

The main and analytical engines reside in a tier called BPM (or simply Server). The main engine controls the execution of processes, their persistence and the firing of events generated by the changes in state of each of the running processes.

The analytical engine has the ability to collect the events generated by the main engine in order to view working statistics in OLAP cubes. Storage in the OLAP server is asynchronous so that the performance of the workflow engine is not degraded.

### 3.4 Connectors tier

The last tier includes a library of connectors that enables users to interact with external systems. The library has connectors to access to relational databases via JDBC, to Java applications via JNI or JMS, to directory services (LDAP) and to applications of the kbee.cms content server. It also includes an API for the development of specific connectors.

## 4. Process Management: Definition

The main objective of Process Management is to assure that the correct activities are performed by the correct person or application in the correct time.

In order to achieve this objective a set of mathematical abstractions are used to enable developers to model processes and manage the execution of tasks.

### 4.1 Cases

A case is the problem to be solved in a specific process. Examples of cases are specific documents; e.g. a file with a specific reference number.

### 4.2 Procedures

A workflow process is designed to manage similar cases by executing tasks in a specific order. A process definition specifies which tasks need to be executed and in what order. Alternative terms of workflow process are "procedure" or "flow chart".

### 4.3 Task

As tasks are executed in a specific order we need to identify the conditions determining the sequence relation between them.

A condition is satisfied or not. Each task has pre-conditions and post-conditions: preconditions should be met in order to allow tasks to be executed, while post-conditions are satisfied after the task has been executed.

### 4.4 Work Items

Many cases can be solved following the same procedure. Thus, a single task can be executed in many cases. For instance, a writing task is executed in several documents. A task that needs to be executed in a specific case is called work item.

Work items are executed by resources. A resource can be either a system or a person with a given role (writer, copy editor, etc.).

## 5. Petri nets for process definition

In order to define processes (workflow) tasks should be specified and given an order of execution.

Petri Nets are a powerful tool for process definition. Petri Nets were invented in 1962 by the mathematician Carl Adam Petri as part of his dissertation at Bonn University.

### 5.1 Classic Petri nets

From Wikipedia:

“A Petri net (also known as a place/transition net or P/T net) is one of several mathematical representations of discrete distributed systems. As a modeling language, it graphically depicts the structure of a distributed system as a directed bipartite graph with annotations.

A Petri net consists of places, transitions, and directed arcs. Arcs run between places and transitions - not between places and places or transitions and transitions. The places from which an arc runs to a transition are called the input places of the transition; the places to which arcs run from a transition are called the output places of the transition.

Places may contain any number of tokens. A distribution of tokens over the places of a net is called a marking. Transitions act on input tokens by a process known as firing. A transition is enabled if it can fire, i.e., there are tokens in every input place. When a transition fires, it consumes the tokens from its input places, performs some processing task, and places a specified number of tokens into each of its output places. It does this atomically, i.e. in one single non-preemptible step.

Execution of Petri nets is nondeterministic. This means two things:

. Multiple transitions can be enabled at the same time, any one of which can fire none are required to fire—they fire at will, between time 0 and infinity, or not at all(!) i.e. it is totally possible that nothing fires at all.

Since firing is nondeterministic, Petri nets are well suited for modeling the concurrent behavior of distributed systems.”

#### 5.1.1 Formal definition

A Petri net is a 5-tuple  $(S, T, F, M_0, W)$ , where

- $S$ , is a set of places.
- $T$ , is a set of transitions.
- $F$ , is a set of arcs known as a flow relation. The set  $F$  is subject to the constraint that no arc may connect two places or two transitions, or more formally:  $F \subseteq (S \times T) \cup (T \times S)$ .
- $M_0 : S \rightarrow \mathbb{N}$  is an initial marking, where for each place  $s \in S$ , there are  $n_s \in \mathbb{N}$  tokens.
- $W : F \rightarrow \mathbb{N}^+$  is a set of arc weights, which assigns to each arc  $f \in F$  some  $n \in \mathbb{N}^+$  denoting how many tokens are consumed from a place by a transition, or alternatively, how many tokens are produced by a transition and put into each place.

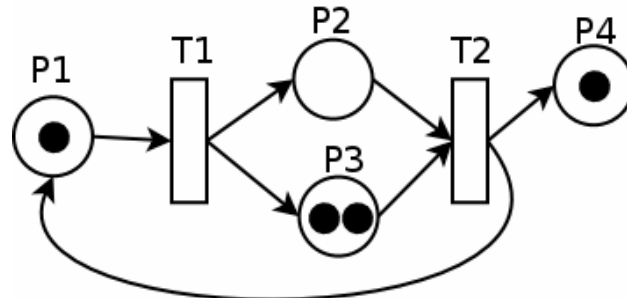


Figura 3. Petri net example

### Basic Description

Explained in words, a standard Petri Net is a graph with the following characteristics:

It is a bipartite graph with two types of nodes called place nodes and transition nodes.

The nodes are connected via directed arcs.

The connection between same types of nodes is permitted.

Pictorially, places are denoted by circles and transitions by rectangles.

At any moment a place may contain one, more or no tokens. (A token is represented with a black spot in the place).

Transitions are the active part of a Petri net as they change the state of the net following these rules:

A transition is enabled when there is at least one token in each input place.

An enabled transition can be fired. When the transition fires, it consumes one token from each input place and produces a token in each output place.

## 5.2 High-level Petri nets

Standard Petri Nets enable the modeling of states, events and conditions; synchronization, parallelism, selections and interactions. However, Petri Nets used for process definition are more extensive and complex.

### 5.2.1 Coloured Petri nets

In some cases, tokens represent objects. To represent the attributes of these objects, the Petri nets model is extended with coloured tokens (or typed tokens).

Each token often has a value often refer to as "color" with a potentially complex structure.

### 5.2.2 Hierarchical Petri nets

As the nets modeling real processes can be large and complex, the standards nets are extended to constructs called 'subnets'.

A subnet is a net connected to the main net through one of its transitions, with the only objective of simplifying and clarifying the specifications in each hierarchical level.

## 5.3 Process Definition with Petri nets

Petri nets used for workflow process definition are called workflow nets

A workflow net meets these two requirements:

- It has a distinguished input place and a distinguished output place.
- Each place and transition in the net contributes in a way to the process of a specific case. In other words: every transition should be located on a path from the input place to the output place.

Workflow tasks are modeled by transitions; conditions are modeled by the places and the case by the tokens.

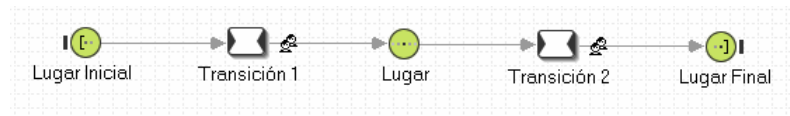


Fig. 1 Petri Net

In short, there exist different and strong reasons for using Petri nets in process definition:

### Formal Semantics

A process specified in terms of a Petri net has a clear and precise definition, because the semantics of the classical Petri net have been formally defined.

### Graphical Nature

Petri nets is a graphical language. Therefore, they are intuitive and easy to learn. Its graphical nature also supports the communication with end users.

### Expressiveness

Petri Nets support all the primitives needed to model a workflow process. All the primitives needed in today's workflow management systems can be modeled.

### Properties

In the last three decades many people have investigated the basic properties of Petri nets. The firm mathematical foundation allows their investigation and development. As a result, there is a lot of common knowledge, in the form of books and articles about them.

### Analysis

There exist several analysis techniques that apply to Petri nets. These techniques can be used to prove properties such as safety, invariance, deadlocks, etc. and to calculate performance measures such as response times, waiting times, occupation rates, etc. In this way it is possible to evaluate alternative workflows using Petri-net-based analysis tools.

### Independent

Petri nets provide a tool-independent framework for modeling and analyzing processes. Moreover, they are not based on a specific software package or in a particular vendor.

## 5.4 Routing Constructs

The routing of cases (specific documents) through the tasks that are to be executed is a key issue in workflow management.

There are four basic routing construct:

### 5.4.1 Sequential routing

It enables the modeling of relations between tasks. Let's take tasks A and B. If task B is executed after the completion of task A, then it is considered that both tasks are executed in a sequential way. Figure 4 shows how sequential routing can be modeled by adding places. Place  $c_2$  denotes the post condition of task A and the precondition of task B.

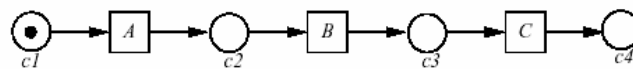


Fig. 4. Sequential routing

Examples of sequential routing are the writing, correction and approval tasks of a document taking place in the context of a publishing workflow. The document is written, then corrected, and eventually approved.

### 5.4.2 Parallel routing

It is used in less strict situations. For example, tasks B and C have to be executed but in an arbitrary execution order.

In order to model this process two additional transitions are added (AND-split and AND-join). They are not only added to accomplish routing functions. The execution of task A AND-split enables tasks B and C and task AND-join D enables after the execution of tasks B and C. Task D synchronize both sub-workflows. As a result, tasks B and C are executed in a parallel way.

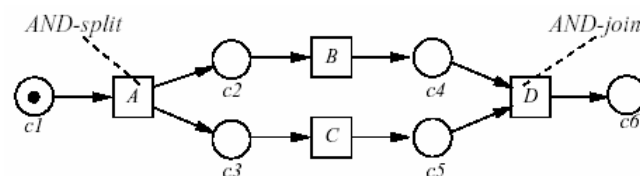


Figura 6. Parallel routing

Examples of parallel routing are the diagramming and writing tasks of the different sections within the editorial workflow of a magazine. Each of a magazine sections are diagrammed and designed both separately and in parallel.

### 5.4.3 Selective Routing

It enables the routing of tasks which may vary depending on the specific case. The variables will depend on the specific attributes of the case, the environment, or the flow of work within the organization where it is carried out.

The selecting routing comprises of two places called OR-split and OR-join to model the selection between two alternatives. An OR-split place may contain multiple output arcs and an OR-join place may contain several input arcs. Figure 7 shows how task A can be followed by task B or task C. The place  $c_2$  denotes the precondition of tasks B and C. However, only one of the tasks can be executed for the token (case) in  $c_1$ .

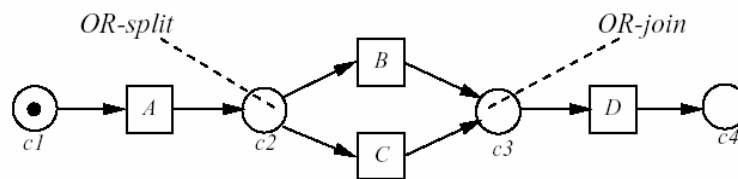


Fig. 7. Selective routing

An example of this type of routing is the detailed level selection needed for a correction task. A written document (task A), can be corrected exhaustively (task B), or quickly (task C) depending on the writer's expertise; once it has been corrected, it is finally approved (task D).

Figure 8 shows another way of modeling a selection between tasks B and C. Transition A comprises of two output places  $c_2$  and  $c_3$ . The transition produces a token in  $c_2$  or in  $c_3$ . The selection between  $c_2$  and  $c_3$  is based on the  $x$  attribute of the token. In figure 5 the selection is carried out the moment task A is completed. In figure 10 the selection is carried out the moment tasks B or C are executed. The term "explicit selection" (explicit OR-split) is used when the selection is based on the workflow attributes. The term "implicit selection" (implicit OR split) is used when the selection is carried out as late as possible.

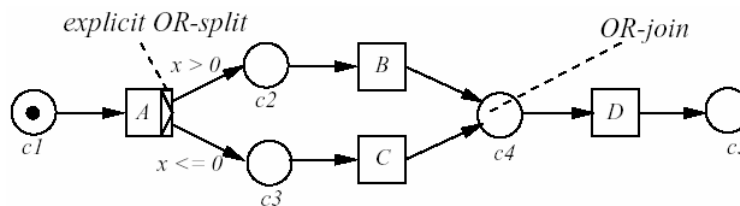


Fig. 8. Selective routing (explicit selection)

Figure 9 shows the possible routing constructs. The constructs AND-split and AND-join depict the usual behavior of a classic Petri net. The constructs implicit OR-split and implicit OR-join are modeled by places. The construct explicit OR-split is modeled as a transition that produces a token in only one of the output places. The construct explicit OR-join is modeled as a transition that can be enabled only if one of its places contains a token.

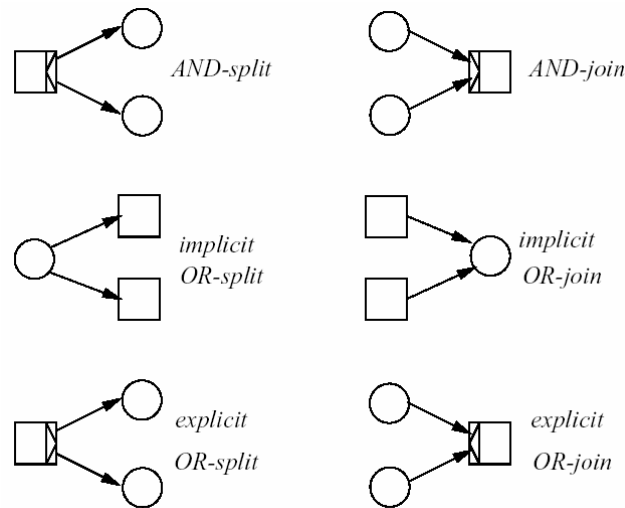


Fig.9. Routing constructions

#### 5.4.4 Iterative routing

An iteration can be modeled with the constructs shown in figure 6. For example, in figure 10, task C is a control task that checks the result of task B. Based on the result of this check, task B may be executed more than once.

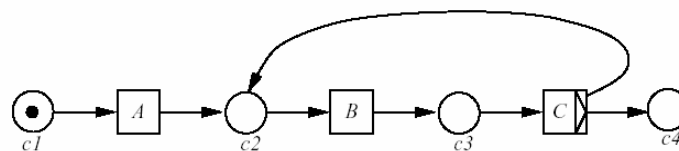


Fig. 10 Iterative routing

Example: the writing tasks in an editorial workflow can be executed once or more times, depending on the result of the approving task.

## 5.5 Triggering

A process definition specifies which tasks should be executed for each state of each specific case. However, it is important to note that if a task can be executed, this does not necessarily imply it should be directly executed. For example, if a task is to be executed by an employee, then the employee has to be available to execute it. If the employee is not available, is on holidays or elsewhere, the task is not executed, and the execution of the task cannot be forced by the workflow engine. Therefore, it is important to differentiate between the enabling of a task (given pre-conditions) and the execution of it.

A trigger is an external condition that fires the execution of an enabled task. The execution of a task instance (activity) for a specific case starts the moment the activity is triggered. However, an activity can only be triggered if the corresponding case is in a state which enables the execution of the task.

We can distinguish between four types of triggers:

### 5.5.1 Automatic triggers

A task is triggered at the moment it is enabled. This type of triggering is used for tasks which are executed by applications or systems which do not require users' interaction.

### 5.5.2 User triggers

A task is triggered by a user who selects it for its execution. This type of triggering is commonly integrated to security services for the distinction of users enabled to activate it.

### 5.5.3 Events triggers

A task is triggered by an external event such as: messages, e-mails, phone-calls, etc.

### 5.5.4 Time triggers

A task is triggered by a clock after the predefined time for the execution of the task is accomplished.

## 6. kbee Workflow Query Language (kbee.WQL)

kbee WQL is a query language with a similar syntax to OQL (Object Query Language). It allows the selection and projection of native attributes or extended attributes of the workflow entities.

The language enables users to query the workflow engine in analogy to queries over a Relational Database Engine.

Users can query upon the process engine variables (Native Models), as well as upon the specific attributes of the problem through the extension of the IT system associated to the processes (Extended Model).

### 6.1 Native and Extended Models

The native model of the workflow comprises of the entities implementing its basic concepts. These are: the processes, activities and work items. Each of these entities contains a set of attributes which are independent from the problem domain where processes are executed, such as the state, timestamp and end time of an activity.

kbee workflow allows extending the model of native entities with entities from the contexts where processes are executed.

Each token circulating through the net contains an extended entity of the native model with a structure described by a set of attributes (colour of the token). These attributes can be added to the attributes of the native entities.

For example, if an extended entity includes an attribute containing a file reference number, this attribute is added to the attributes of the activities executed upon said entity.

Therefore, the user has the possibility of querying upon an activity not only the timestamp or end time of it, but also the reference number of the file upon which it is acting.



## 6.2 kbee WQL language

### Sintaxis

kbee WQL is a query language with a similar syntax to OQL (Object Query Language). It allows the selection and projection of native attributes or extended attributes of the workflow entities. A language sentence has the following structure:

```
SELECT
activity | workitem | process | attribute 1, attribute2 ..., attribute n |
count
[FROM
activity | process | workitem ]
[WHERE
search condition 1
[AND search condition 2 .... ]
[OR search condition 3 .... ]
[GROUP BY attribute 1, attribute 2 ... , attribute n]
[ORDER BY attribute 1, attribute 2 ... , attribute n [desc] ]
[OFFSET number ]
[LIMIT number ]
```

### Operators

The operators used for search conditions can be the following:

Operator	Description
=	Equal to
≠	Not equal to
>	Greater than
<	Less than
>=	Greater than or Equal to
<=	Less than or Equal to
Func	Operador funcional unitario

### Native Attributes

The following is a complete list of native attributes which can be projected and evaluated in a condition:

#### Selection of Entities

One of the basic uses of kbee WQL is the selection of entities which meet the required conditions.

Let's consider the following sentence:

```
Select Activity Where expediente = '23454'
```

This sentence selects all the activities applied on file '23454'. Please notice that 'Expendiente' is an extended attribute of the native model.

#### Projection of attributes

It also allows projecting only the attributes of the selected entities. For example:

```
Select task, state From Activity Where expediente = '23454'
```

This sentence selects the task and state of activities that have been applied upon the same file of the previous case.



### Functions

As part of a query conditions, developers can apply some functions to evaluate the state of entities. For example, the function "isEnabledFor" evaluates upon a work item if the user parameter is enabled or not to take it. The following sentence is an example of this function:

```
Select Workitem Where isEnabledFor:(user)
```

This sentence selects all the work items enabled to the user parameter of name "user".

### Counters

The count function is very useful to calculate indicators. For example, the following sentence calculates an indicator upon which an alarm can be activated if the obtained number is greater than zero:

```
Select count From Activity Where expediente='2354' And state='Error'
```

### Other properties

kbee WQL also includes functions for grouping and paging results.

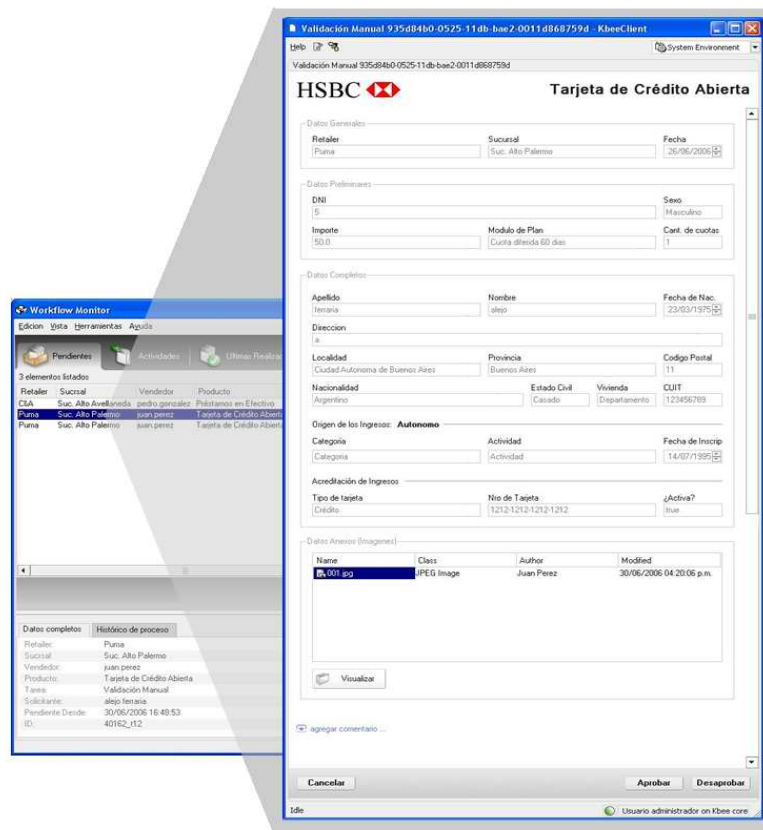


Figure 11. Console for process monitor and tasks consulting built in a high-level language (C++) and using kbee.WQL to be integrated to the workflow engine.



## 7. Analytical server and OLAP reports

kbee.workflow enables the asynchronous logging of each change in state of a running process in a database integrated to an OLAP server.

		Producto						
Fecha	Retailer	Todos	Préstamos de Consumo	Préstamos en Efectivo	Tarjeta de Crédito Abierta	Tarjeta de Crédito Cerrada	Tarjeta de Crédito Semi Abierta	
Todas	Todos	\$5,000.00	\$500.00	\$2,700.00	\$400.00	\$300.00	\$1,100.00	
	CBA	\$3,200.00	\$500.00	\$2,700.00				
	Suc. Alto Avellaneda	\$3,200.00	\$500.00	\$2,700.00				
	Puma	\$1,800.00			\$400.00	\$300.00	\$1,100.00	
2006	Todos	\$5,000.00	\$500.00	\$2,700.00	\$400.00	\$300.00	\$1,100.00	
	CBA	\$3,200.00	\$500.00	\$2,700.00				
	Suc. Alto Avellaneda	\$3,200.00	\$500.00	\$2,700.00				
	Puma	\$1,800.00			\$400.00	\$300.00	\$1,100.00	
06/2006	Todos	\$5,000.00	\$500.00	\$2,700.00	\$400.00	\$300.00	\$1,100.00	
	CBA	\$3,200.00	\$500.00	\$2,700.00				
	Suc. Alto Avellaneda	\$3,200.00	\$500.00	\$2,700.00				
	Puma	\$1,800.00			\$400.00	\$300.00	\$1,100.00	

kbee.OLAP server (based on Open Source Mondrian software) contains a Java-based engine which executes written queries in the Standard MDX created by Microsoft in 1998 (Multi Dimensional eXpressions), takes information from a Relational Database (typically, one or more pivot tables where each of the dimensions has an entry), and displays the result in a multidimensional format using an API Java.

The data shown by kbee.OLAP is taken from said Database which is asynchronously updated after each change in state of the processes executed by the workflow engine. Both OLAP and the Database can be run in another server, in order not to degrade the performance of the management applications.

With kbee.OLAP, developers can define the variables (dimensions) of the business, and make complex statistical queries without programming (It allows non-specialized users make queries by specifying the dimensions on rows and columns, plus the necessary selection criteria and filters).

kbee.OLAP enables developers to predefine a great number of commonly asked queries and make them available for users, so they can be easily accessed.

The multidimensional OLAP cube can seamlessly integrate other sources of information, acting as an integral tool for analytical report and business intelligence.



## Appendix I

### kbee.workflow license

kbee.workflow is delivered with LGPL 2.1 license. The English version of said license is included in this appendix, and it can be also found in <http://www.gnu.org/licenses/lgpl.html>,

#### GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

#### Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library. To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by



obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License").

Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative



work Under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language.

(Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it.

For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.



Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables



containing this object code plus portions of the Library will still fall under Section 6.) Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications. You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy. For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a



combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
- b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library 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.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.  
You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to



the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public 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. Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.