# Web-Enabled Workflow Management System

# (DPSFLOW)

Undertaken by
**Khurram Shahzad**

Supervised by
**Prof. Dr. Khalid Rashid**
Dean Faculties of Management Sciences &
Applied Sciences

**Department of Computer Science**
**Faculty of Applied Sciences**
**International Islamic University, Islamabad**
**(2004)**

بسم الله الرحمن الرحيم

**International Islamic University Islamabad**
**Department of Computer Science**

Dated: 3. 4. 2004

**FINAL APPROVAL**

It is certified that we have read the project report submitted by Khurram Shahzad (34-CS/MS/01) and it is our judgment that this project is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the Masters Degree in Computer Science.

**External Examiner**
Mr. Shaftab Ahmed
Senior Faculty Member
Department of Computer Science
Behria University Islamabad.

[illegible]

**Internal Examiner**
Dr. S. Tauseef-ur-Rehman
Head Dept. of Telecommunication Engineering,
Faculty of Applied Sciences
International Islamic University Islamabad.

[illegible]

**Supervisor**
Prof. Dr. Khalid Rashid
Dean, Faculties of Management Sciences
& Applied Sciences,
International Islamic University, Islamabad.

[illegible]

A theses submitted to the

**DEPARTMENT OF COMPUTER SCIENCE**

**INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD**

as a partial fulfillment of the requirements

for the award of the degree of

MS (Computer Science)

**Dedicated to**

**Those who are**

**Lucky in**

**this world as well as in the**

**“Life after Death”**

## DECLARATION

I hereby declare that this project report, neither as a whole nor as a part thereof has been copied out from any source. It is further declared that I have completed my work on the basis of my personal efforts and under the sincere guidance of my teachers. If any part of this work is proved to be copied out from any source or found to be reproduction of someone else, I shall standby the consequences. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

**Khurram Shahzad**
**34-CS/MS/01**

## ACKNOWLEDGENENT

It is the blessings of Allah Almighty who gave me courage to accept and complete this project. I am very thankful to Prof. Dr. Khalid Rashid who provided full assistance in my work. I owe a great deal to all my friends who extended to me whatever help I needed. Also I thank my parents for their undying support.

**Khurram Shahzad**
**34-CS/MS/01**

# PROJECT IN BRIEF

| | |
|---|---|
| **Project Title:** | DPSFLOW (Web-Enabled Workflow Management System) |
| **Organization:** | Digital Processing Systems, Inc. |
| | International Islamic University, Islamabad. |
| **Undertaken By:** | Khurram Shahzad |
| **Supervised By:** | Professor Dr. Khalid Rashid |
| | Dean, Faculties of Management Sciences & Applied Science, |
| | International Islamic University, Islamabad. |
| **Tool Used:** | Sun java Development Kit 1.4 & Sun J2EE APIs |
| | Borland JBuilder 8.0.06 Enterprise Edition |
| | PLATINUM ERWin 3.5.2 |
| | Web Server Apache Tomcat |
| | Oracle 8i Data Base Server |
| **Technology Used:** | Java 2 Standard Edition (Java Swing), Java 2 Enterprise Edition (JDBC, Servlets, jsp) |
| **Operating System:** | Platform independent |
| **System Used:** | IBM Compatible |
| **Date Started:** | September, 2002. |
| **Date Completed:** | December, 2003. |

# Abstract

This thesis develops a methodology for a new light weight HTTP enabled workflow management system, called DPSFLOW, and entirely based on the Java and WWW as its basic technologies. In DPSFLOW Java is used as the modeling (build time) as well as the implementation language for workflow enactment. Modular and extendible workflow definitions are possible due to the object-orientation of Java. Using WWW and Java eases the implementation effort of the workflow engine, since HTTP and the Java API already include a lot of functionality, and execution of workflows in the workflow engine that is basically combination of the Application Server and Database Server. It also includes a build time and a run time dimension of workflow imposing different requirements. The build time requirements comprise all issues related to modeling of control and data flow. Most of these requirements are not specific to workflow modeling, like graphical or formal representation, ease of use, readability, and modularity, or correctness.

The run time requirements are related to the activation, enactment and termination of workflows. The system shall be extendible to be able to handle future unforeseen application scenarios. Continuously growing application areas and increasing organizational coverage call for a scalable system. The system should be open that it can run in a distributed.heterogeneous computing infrastructure in future

# TABLE OF CONTENTS

# Chapter 1
# INTRODUCTION

# 1. Introduction

Business processes which are optimized in time and flow are becoming crucial for the commercial success of companies. Such processes can be modeled as a composition of activities and subsequently mapped onto workflow management systems.

A workflow represents the operational aspects of a work procedure: the structure of tasks and the applications and humans that perform them; the order of task invocation; task synchronization and the information flow to support the tasks; and the tracking and reporting mechanisms that measure and control the tasks.

A workflow management system completely defines manages and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic.

## 1.1 An activity based workflow management system

Being activity based workflow management system means that the processes or the workflows are made of activities to be completed in order to get something done. This differs from entity based workflows, where the focus is set on a given document

For example, the process describing a research in a campus library can be handled as an activity based process: we have to get a signed authorization from the teacher, find the books of interest using either the computer catalogs or the librarian's help, and then check-out the books we need (signing the appropriate stuff). This is a list of activities to be carried out in order to get books out of the library. There is no document attached in this process.

Another example of an activity based process might be represented by the fund request submission process. A teacher of the university has to submit a request for buying a new computer. The process starts by filling in a form where the teacher specifies what he intends to buy and how much does it cost. The form will be handled by the administration office and, if the total cost goes beyond a given threshold, it requires the personal authorization by the administration chief. Budget is then checked, to see if the teacher has appropriate funds left for the acquisition. At the end an e-mail is sent to the teacher informing about the result of the submission; the request is filed in the request archive and the database is updated subtracting the requested cost from the available money. This process has a main document (the request) but it has a lot of side actions to be carried out as well: signed authorization, archive insertion, database update, e-mail sending. The best way to describe this is through an activity based workflow, where the process is made of activities to be carried out.

On the other hand simpler processes do not require the complex structure or an activity based workflow management system, and a simple entity-based workflow would suffice. For example, publication of documents on a web site can be simply modeled by the given document going through the states of new, submitted, and then approved or rejected. In these kinds of workflows the document is the main issue and its available actions are defined by its current state.

## 1.2 Main issue

The main issue for a workflow management system is answering the question "who must do what, when and how". In an activity based workflow management system (as DPSFLOW) this question finds an answer.

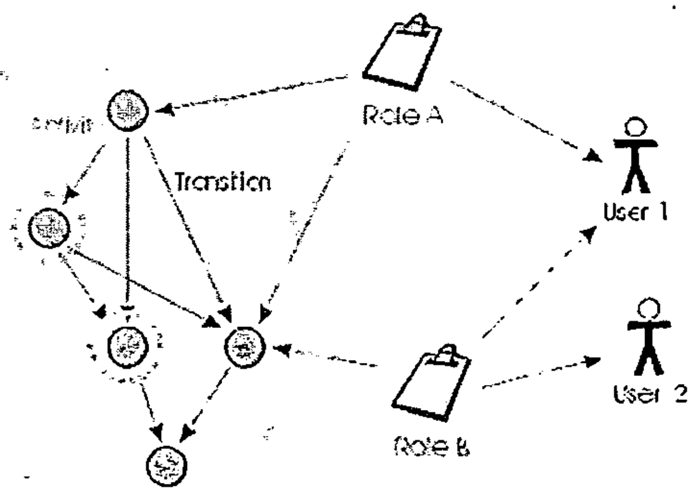


Fig. 1.1 Simple Workflow Model

The process defining the sequence of activities to be carried out says what should be done and when by the definition of activities and transition in stand alone editor. An activity (the what part of the issue) represents something to be done: giving authorization, updating a database, sending an e-mail, loading a truck, filling a form, printing a document etc. Transitions define the appropriate sequence of activities for a process (the when part of the issue).

Each activity will have an associated attributes designed to carry out the job: the "*how*" part.

The "*who*" part is determined by the user assigned to carry out the activity, through its application. Usually activity applications will be used by someone, a person, but in many other cases the entity assigned to an activity might as well be an automatic system. After all, why should we have a person waste its time doing a database update that could be done by a much faster computer?

## 1.3 Benefits of DPSFLOW

One benefit to use a workflow management system is to improve the efficiency and performance of the processes we usually handle. A lot of activities we usually do can actually be carried out by an automatic system. Database update, research, document archiving and so on are activities that a computer will carry out without human intervention. This means that the activity job will be carried out much faster and the human resources can be used in some more valuable way (and they will be grateful for that).

Another benefit to use a workflow management system is to always have the answer to the question "*who must do what, when and how*". Formalizing our process means wasting no time in deciding what to do, having the right person doing the job and being sure that our issue will have a way to be completed appropriately.

## 1.4 The Workflow definition

The goal of a workflow definition is to give answer of the question "who must do what, when and how". Activities and transitions describe the workflow we want to model (what and when part). Applications describe the how part and they are associated to activities. Users and roles describe the who part.

### 1.4.1 Application Example

An example of a simple business process is explained in this section to illustrate the usability of DPSFLOW. Figure 1.2 illustrates the composite workflow *"purchasing new computer equipment"* which could happen in a company. It involves five workflow participants: a manager, a computer specialist, a financial specialist, a secretary and an auto-user. The scenario is as follows. The manager realizes that new computer equipment is needed. He contacts the computer specialist to get informed about possible shops and cost. He simultaneously asks the financial specialist to look for the current budget being available. If the budget is higher than the cost, the secretary will order the equipment from the chosen shop. The manager gets the message of a successfully given order. Otherwise, if the budget is too low, he gets an appropriate trouble message. Joining the parallel tasks of the specialists and the decision how to continue, is defined as an auto-user activity, i.e. it is executed automatically.

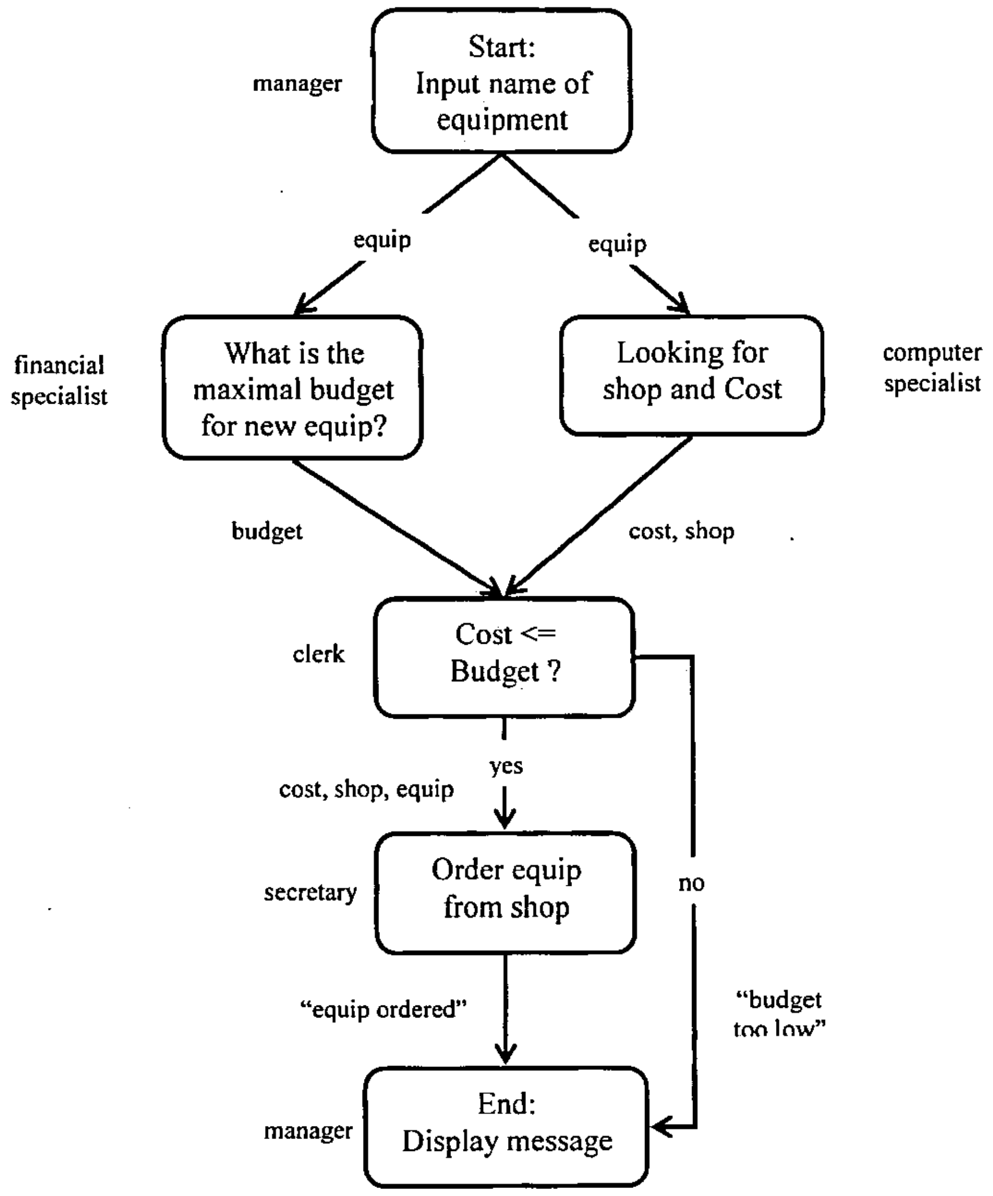


Fig. 1.2 Simple Business Process Model

# Chapter 2

# PROBLEM DEFINITION AND PROPOSED SYSTEM

# 2. Problem Definition and Proposed System

The state of the art in workflow management has been determined so far by the functionality provided in commercial systems. Many products were developed without a clear understanding of the user requirements and these products were quite unprepared to meet the demands placed upon them by eager users.

Issues such as performance, scalability or reliability are hardly ever considered in these areas, an unfortunate characteristic inherited by workflow products. Usually commercial workflow products are not based on OLTP (On-Line Transaction Processing) or database technology. They use file system as transaction storage. Most of the time user activity states are kept in live threads and managed via programming. As a consequence, the robustness and technological maturity is lacking in workflow systems. Most glaring limitations of existing systems are:

## 2.1 Client/Server Model – A Major Limitation

From an architectural perspective workflow management systems are mostly client/server systems. A centralized workflow engine coordinates the workflow clients being related to the workflow participants.

The growth of the internet and especially the World Wide Web (WWW) has caused an increasing shift from dedicated client/server software towards the exploitation of WWW technology for many kinds of applications, also for workflow management systems.

Independence of operating system and hardware platform at the client side is guaranteed when using the standard WWW protocols and document formats. The basic WWW applications still follow the client/server paradigm.

## 2.2 The Architectural Limitations

The architectural limitations (single database, poor communication support, lack of foresight in the designs, the problems posed by heterogeneous designs) have prevented existing systems from being able to cope with heavy load of Workflow participants.

## 2.3 Lack of Robustness

One of the major limitations of existing systems is their lack of robustness and limited availability. Current products have a single point of failure (the database) and no mechanism for backup. This is not as much a flaw as a design decision, since these products were initially intended for small groups and small loads. Very large workflow management systems will involve several thousand users, hundreds of thousands of concurrently running processes and several thousand sites distributed over wide area networks. There will be critical systems and, as such, their continuous availability is crucial, in the same way continuous availability is the key to many banking and corporate database applications. It is not reasonable to expect corporations to rely on a workflow management if a single database failure can bring the entire system to a halt. Moreover, since workflow systems will operate in large distributed and heterogeneous environments, there will be a variety of components involved in the execution of a process. Any of these components can fail and, nowadays, there is not much that can be done about it. Existing systems lack the redundancy and flexibility necessary to replace failed components without having to interrupt the functioning of the system.

## 2.4 Proposed System

We propose a workflow management system, called DPSFLOW, entirely based on the WWW and Java as its basic technologies. In DPSFLOW Java is used as the build time (modeling) language as well as the implementation language for workflow enactment. Modular and extendible workflow definitions are possible due to the object-orientation of Java. Using WWW and Java eases the implementation effort of the workflow engine, since HTTP and the Java API already include a lot of functionality, and execution of workflows in the workflow engine that is basically combination of the Application Server and Database Server.

## 2.5 Goals

1. Have a **Workflow Engine** that will be able to:
   - understand workflow definitions
   - handle activity instances and their flowing in the workflow
   - return work lists to users assigned to activities

2. Have a **Process Designing Tool**:
   - intuitive and easy to use
   - able to design complex processes

3. Have a **Web Oriented User Interface** for a greater probability of use on any platform. The entire application programming interface will be callable via http

# Chapter 3

# SYSTEM ANALYSIS

# 3 System Analysis

Analysis is the software engineering task that bridges the gap between system-level software allocation and software design. Requirments analysis enables the system engineer to specify software funvtion and performance, indicate software's interface with other system elements, and establish constraints that software must meet. Most of the time spent in project development is given to analysis. System analysis was done using prototyping and object oriented methodology.

## 3.1 Functional requirements

Functional requirements describe the functions or functional behavior of the system. Functional requirements describe the functions according to the user point of view, how user interacts with the system and how the system behaves with the user.

## 3.2 General Requirements

The user (Administrator/manager) will first draw the workflow processes map in IDE provided by the system. Then he will Assign activities to users. These users will be the names like clerk, manager, director etc. i.e.; the persons involved in any business transaction that we want to automate using workflow management system.

On the successful creation of workflow process map it will be brought on to the web by special means. Using Web interface the administrator will assign database roles to the users created above. User management i.e.; password changing etc will be done from database end. Moreover administrator will monitor the Workflow in all its parts. At the lower end there will be a database server that will hold all the workflows or process maps build. There will be no live thread that will hold the information. All the information will be persistent.

The user that is a part of business process or workflow process will be provided web based graphical user interface. It will be created dynamically, based on the workflow process map created by the administrator. Definitely to build graphical user interface for the specific user, the workflow process map will be filtered. We will model this complex application using UML. Defining a model makes it easier to break up a complex application or a huge system into simple, discrete pieces that can be individually studied. We can focus more easily on the smaller parts of a system and then understand the "big picture." The reasons behind modeling can be summed up in two words:

- Readability
- Reusability

## 3.3 System Considerations

The following are major considerations that we would see carefully.

- Visualization tool (Workflow Builder) to develop processes.
- Process Definition using Workflow Builder
- Eliminate the record keeping of date stamping, logging, and tracking work.
- Cut out the effort looking for particular items.
- Ability to suspend work.
- Running parallel processes.

## 3.4 Information Description

The first operational analysis principle requires an examination of the function domain. The information domain contains three different views of the data and control as each is processed by a computer program: (1) information content and relationships, (2) information flow, and (3) information structure

### 3.4.1 Information Content Representation

Information content represents the individual data and control objects that comprise some large collection of information that is transformed by the software.

To understand this work flow, following are the terminologies which will be discussed in following chapters.

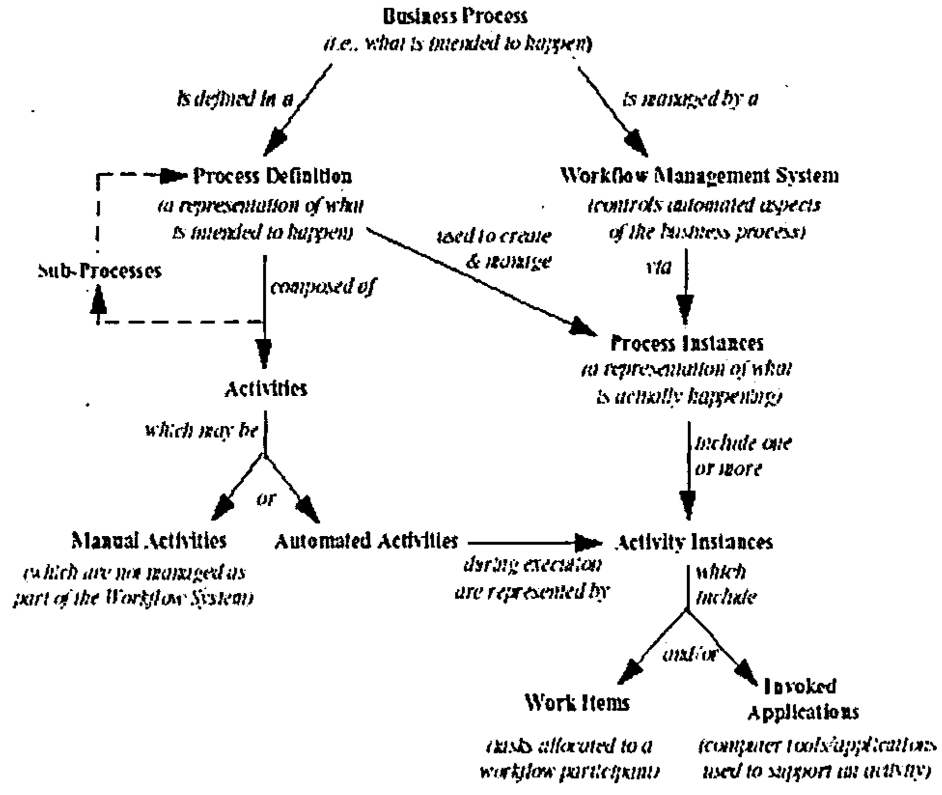

Business Process
(i.e., what is intended to happen)
is defined in a
is managed by a
Process Definition
(a representation of what is intended to happen)
Workflow Management System
(controls automated aspects of the business process)
used to create & manage
via
Sub-Processes
composed of
Process Instances
(a representation of what is actually happening)
Activities
which may be
include one or more
or
Manual Activities
(which are not managed as part of the Workflow System)
Automated Activities
during execution are represented by
Activity Instances
which include
and/or
Work Items
(tasks allocated to a workflow participant)
Invoked Applications
(computer tools/applications used to support an activity)

Fig. 3.1 Workflow Information Contents Hierarchy

## 3.4.2 Information Flow Representation

Information flow represents the manner in which data and control change as each moves through the system. Macro level flow of control of workflow management tool is given below which explains the architecture of Workflow components.

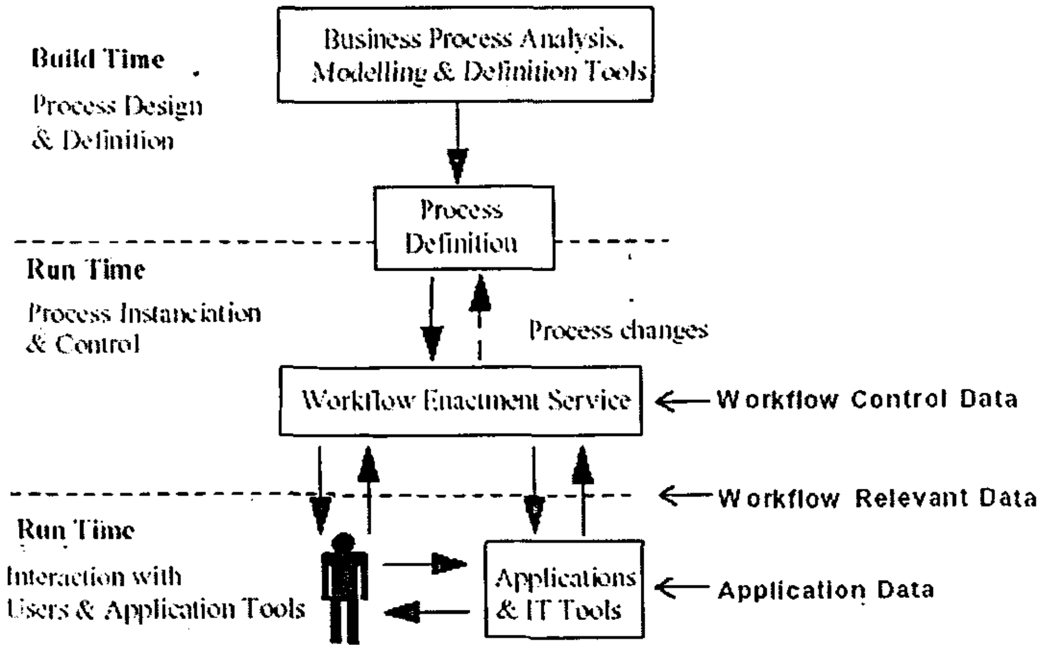

Build Time
Process Design & Definition
Business Process Analysis, Modelling & Definition Tools
Process Definition
Run Time
Process Instanciation & Control
Process changes
Workflow Enactment Service
Workflow Control Data
Workflow Relevant Data
Run Time
Interaction with Users & Application Tools
Applications & IT Tools
Application Data

Fig. 3.2 Workflow Information Flow Hierarchies

## 3.5 Workflow Components and their Explanation

We now explain in detail all components that are used in Workflow Management Systems. Without understanding these components we cannot further proceed.

### 3.5.1 Workflow

**Definition**

The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

***Synonyms***

- Workflow Management
- Workflow Computing
- Case Management

### 3.5.2 Workflow Management System

**Definition**

A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications.

***Synonyms***

- Workflow Automation
- Workflow Manager
- Workflow Computing System
- Case Management

### 3.5.3 Business Process

**Definition**

A set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.

***Synonyms***

- Process

### 3.5.4 Activity

**Definition**

A description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resources(s) to support process execution; where human resource is required an activity allocated to a workflow participant.

***Synonyms***

- Step
- Node
- Task
- Process Element
- Operation
- Instruction

### 3.5.5 Automated Activity

**Definition**

An activity which is capable of computer automation using a workflow management system to manage it during execution of the business process of which it forms a part.

***Synonyms***

- Workflow Activity
- Activity

### 3.5.6 Manual Activity

**Definition**

An activity within a business process which is not capable of automation and hence lies outside the scope of a workflow management system. Such activities may be included within a process definition, for example to support modeling of the process, but do not form part of a resulting workflow.

**Synonyms**

- Non-automated Activity
- Manual Step
- Human Task
- Manual Work

### 3.5.7 Instance

**Definition**

The representation of a single enactment of a process, or activity within a process, including its associated data. Each instance represents a separate thread of execution of the process or activity, which may be controlled independently and will have its own internal state and externally visible identity, for example, to record or retrieve audit data relating to the individual enactment.

### 3.5.8 Process

**Definition**

A formalized view of a business process, represented as a coordinated (parallel and/or serial) set of process activities that are connected in order to achieve a common goal.

*Usage*

- Example: An eight activity process

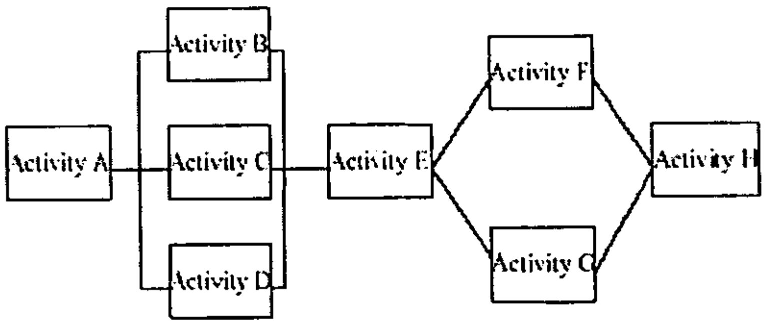


Fig. 3.3 Processes in abstract form

***Synonyms***

- Activity Network
- Directed Graph
- Petri Net
- Model
- Instruction Sheet

## 3.5.9 Process Instance

**Definition**

The representation of a single enactment of a process.

***Synonyms***

- Process Definition Instance
- Case
- Workflow Definition Instance
- Instruction Sheet Instance

## 3.5.10 Activity Instance

**Definition**

The representation of an activity within single enactment of a process, i.e. within a process instance.

***Synonyms***

- Step Instance
- Node Instance
- Task Instance
- Work Element Instance

### 3.5.11 Workflow Participant

**Definition**

A resource which performs the work represented by a workflow activity instance. This work is normally manifested as one or more work items assigned to the workflow participant via the wordlist.

***Synonyms***

- Actor
- User
- Role Player
- Work Performer

### 3.5.12 Sub Process

**Definition**

A process that is enacted or called from another (initiating) process (or sub process), and which forms part of the overall (initiating) process. Multiple levels of sub process may be supported.

***Synonyms***

- Subflow
- Sub Workflow

### 3.5.13 Process Execution

**Definition**

The time period during which the process is operational, with process instances being created and managed.

***Synonyms***

- Process Enactment
- Run Time Operation
- Workflow Execution (strictly this refers only to the automated parts of process execution)

### 3.5.14 Parallel Routing

**Definition**

A segment of a process instance under enactment by a workflow management system, where two or more activity instances are executing in parallel within the workflow, giving rise to multiple threads of control.

***Synonyms***

- Parallel workflow processing
- Concurrent Processing

### 3.515. Sequential Routing

**Definition**

A segment of a process instance under enactment by a workflow management system, in which several activities are executed in sequence under a single thread of execution. (No AND-Split or AND-Join conditions occur during sequential routing.)

***Synonyms***

- Serial Routing

### 3.5.16 AND-Split

**Definition**

A point within the workflow where a single thread of control splits into two or more threads which are executed in parallel within the workflow, allowing multiple activities to be executed simultaneously

*Usage*

- At an And-Split separate threads of control within the process instance are created; these threads will proceed autonomously and independently until reaching an And- Join condition.
- **Example**

In certain workflow systems all the threads created at an And-Split must converge at a common And-Join point (Block Structure); in other systems convergence of a subset of the threads can occur at different And-Join points, potentially including other incoming threads created from other And-split points. (Free Graph Structure)

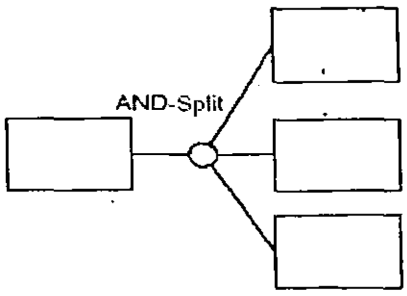


Fig. 3.4 AND- Split

### *Synonyms*

••Split

## 3.5.17 OR-Split

### Definition

A point within the workflow where a single thread of control makes a decision upon which branch to take when encountered with multiple alternative workflow branches

***Usage***

••An OR-Split is conditional and the (single) specific transition to next activity is selected according to the outcome of the Transition Condition(s).

••Example

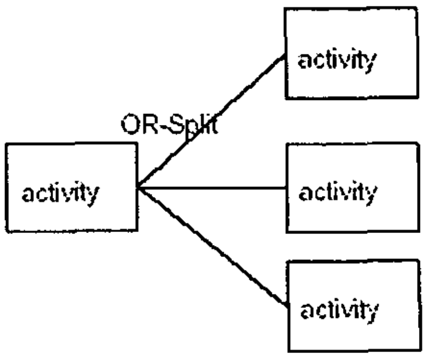


Fig. 3.5 OR- Split

***Synonyms***

- Conditional Branching
- Conditional Routing
- Switch
- Branch

## 3.6 Users of DPSFLOW

Users involved in DPSFLOW are:

### 3.6.1 Workflow Manager

This actor is responsible for the workflow top level management. This use cases include:

- Draw process maps
- Assign activities to users
- Monitor the Workflow in all its parts
- Exception handling and redesign the existing workflow

### 3.6.2 User

This actor is responsible execution of a set of assigned activities. This use cases also include:

- Get the to-do list of all the activities
- Perform his activity according to the dynamically loaded workflow process

## 3.7 Use Case in Expanded Format

### 3.7.1 Create Node

| | |
|---|---|
| **Use Case ID:** | <<UC-CN-01>> |
| **Use Case Name:** | Create Node |
| **Date Created:** | Sep,2002 **Date Last Updated:** Feb,2003 |
| **Actor:** | Any person who is a member of the administrative group/role |
| **Description:** | When actor wants to create new workflow, he first has to drag node component from tool bar, then place it on the client area. A dialog box will be opened, i.e, use case <<UC-CM-02>> will be executed. |
| **Preconditions:** | 1. User has the privileges of Administrative group/role and Workflow Builder has been loaded successfully. |
| **Priority:** | High |

**Typical Course of Events:**

| Actor Actions | System Response |
|---|---|
| 1. The user selects Node Button from tool bar by pressing it. | 2. Button status is shown pressed and arrow pointer changes. |
| 3. User clicks in the client area to place node | 4. A dialog box is opened and the use case <<UC-CM-02 >> is called |

### 3.7.2 Create Activity

| | |
|---|---|
| **Use Case ID:** | <<UC-CA-02>> |
| **Use Case Name:** | Create Activity |
| **Date Created:** | Sep,2002 **Date Last Updated:** Feb,2003 |
| **Actor:** | Any person who is the member of the administrative group/role |
| **Description:** | When actor has crated node, this use case will be executed. User will be prompted to create activity. A activity will have attributes. |
| **Preconditions:** | 1. Use case <<UC-CN-01>> has been executed. i.e., user has the created the node. |
| **Priority:** | High |

**Normal Course of Events:**

| **Actor Actions** | **System Response** |
|---|---|
| 1. The user fills the options on the dialog box and press OK. | 2. Validates options on OK button press.<br>3. A Hash Map or Linked List will be generated containing temporary activity and message.<br>4. System places icon on designer panel, representing activity. |

## 3.7.3 Assign Activities

| | |
|---|---|
| **Use Case ID:** | <<UC-AA-03>> |
| **Use Case Name:** | Assign Activities (to users) |
| **Date Created:** | Sep,2002 **Date Last Updated:** Feb,2003 |
| **Actor:** | Any person who is the member of the administrative group/role |
| **Description:** | During the crated of node use case, this use case will be executed. User will be given option to enter the name the user for current activity. An activity will have exactly one user assigned. |
| **Preconditions:** | 1. Use case <<UC-CN-01>> has been executed. i.e., user has the created the node. |
| **Priority:** | High |

| **Normal Course of Events:** | **Actor Actions** | **System Response** |
|---|---|---|
| | 1. The user fills the options on the dialog box and presses OK. | 2. Validates options on OK button press.<br>3. A Linked List will be generated containing activity users and their messages. |

### 3.7.4 Delete Activity

| | | | |
|---|---|---|---|
| **Use Case ID:** | <<UC-DA-04>> | | |
| **Use Case Name:** | Delete Activity | | |
| **Date Created:** | Sep,2002 | **Date Last Updated:** | Feb,2003 |
| **Actor:** | Any person who is the member of the administrative group/role | | |
| **Description:** | When actor has selected node and after right clicking he selects "Delete" option then this activity will be performed. | | |
| **Preconditions:** | 1. Use case <<UC-CN-01>> has been executed. i.e., user has the created the node. | | |
| **Priority:** | Medium | | |
| **Normal Course of Events:** | **Actor Actions** | **System Response** | |
| | 1. The user selects any node and then presses delete button or clicks (X) button on the tool bar. | 2. The system will check dependency <<UC-CN-01>>of selected node. | |
| | | 3. Deletion confirmation dialog will be prompted. On confirmation that node will be deleted from the client view panel. | |

### 3.7.5 Create Relation

| | | |
|---|---|---|
| **Use Case ID:** | <<UC-CR-05>> | |
| **Use Case Name:** | Create Relation | |
| **Date Created:** | Sep,2002 **Date Last Updated:** Feb,2003 | |
| **Actor:** | Any person who is the member of the administrative group/role | |
| **Description:** | When want to link two nodes, this use case will be executed. User will be prompted to select relation type. | |
| **Preconditions:** | 1. Use case <<UC-CN-01>> has been executed. i.e., user has the created the node. | |
| **Priority:** | High | |
| **Normal Course of Events:** | **Actor Actions** | **System Response** |
| | 1. The user selects RELATION icon from toolbar | 2. System changes button appearance to press. |
| | 3. The user presses mouse left button down on any node created in <<UC-CN-01>>. | 4. System selects that node, updates link list. |
| | 5. The user drags mouse directing destination node and releases mouse. | 6. System creates arrow, System opens popup menu giving option to select transfer condition for this relation. |
| | 7. The user selects any option from transfer condition. | 8. System sets arrow label, sets "Source Node" attributes, updates linked list. |

## 3.7.6 Save Model

| | | |
|---|---|---|
| **Use Case ID:** | <<UC-SM-06>> | |
| **Use Case Name:** | Save Model | |
| **Date Created:** | Sep,2002 | **Date Last Updated:** Feb,2003 |
| **Actor:** | Any person who is the member of the administrative group/role | |
| **Description:** | When user wants to save workflow in database, this use case will be executed. User will be prompted to select name and location of the database. | |
| **Preconditions:** | 1. Use case <<UC-CN-01>> has been executed. i.e., user has the created the node. | |
| **Priority:** | High | |
| **Normal Course of Events:** | **Actor Actions** | **System Response** |
| | 1. The user selects save option from File menu and presses save button. | 2. System validates the workflow created and after validation it pops the save dialog box. |
| | 3. User fills-in the Save Dialog, i.e.; user provides workflow name and version and presses the save button. | 4. System opens the connection with the data base and then parses all linked lists and saves Workflow in the database. |

# Chapter 4

# SYSTEM ARCHITECTURE & DESIGN

# 4 System Architecture & Design

The purpose of design is to create architecture for the evolving implementations. Object oriented design is a method of designing that encompasses the process of object oriented decomposition and a notation for depicting logical and physical as well as static and dynamic models of system under design.

The designed phase focuses on defining the software to implement the application. The design object is to produce a model of the system which can be used later to build the system. The designed goal is to find the best possible design within the limitations imposed by the requirement and physical social environment in which the system will operate.

## 4.1 Activity diagrams

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel.

Activity diagrams should be used in conjunction with other modeling techniques such as interaction diagrams and state diagrams. The main reason to use activity diagrams is to model the workflow behind the system being designed. Activity Diagrams are also useful for: analyzing a use case by describing what actions need to take place and when they should occur; describing a complicated sequential algorithm; and modeling applications with parallel processes.

However, activity diagrams should not take the place of interaction diagrams and state diagrams. Activity diagrams do not give detail about how objects behave or how objects collaborate.

Following are the major Activity Diagrams of DPSFLOW System.

### 4.1.1 Initialize Workflow Builder

When we start Workflow Builder then these processes are done, which are shown in following activity diagram.

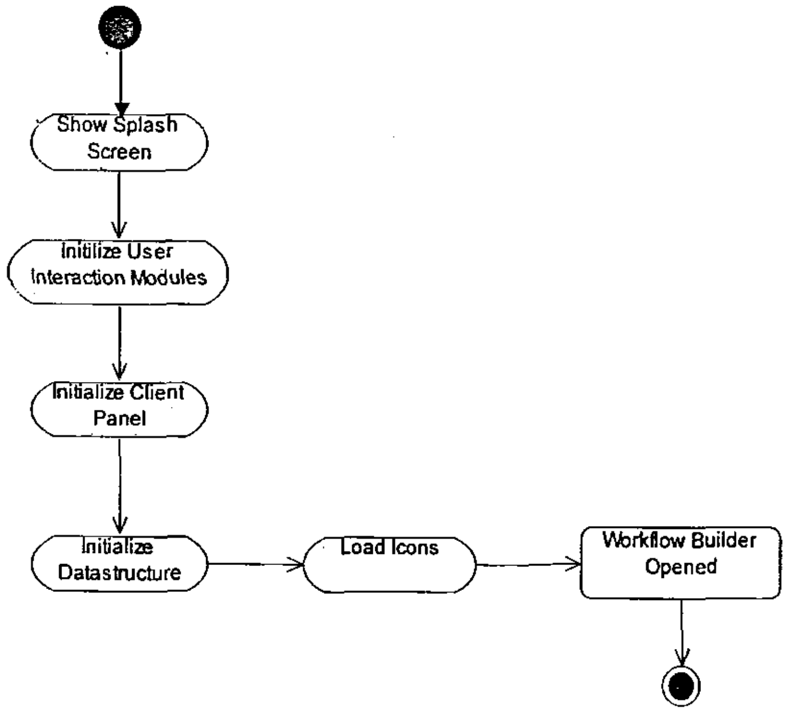


Fig. 4.1 Initialize Workflow

### 4.1.2 Create Workflow

When user wants to create a new Workflow then following activities are accomplished.

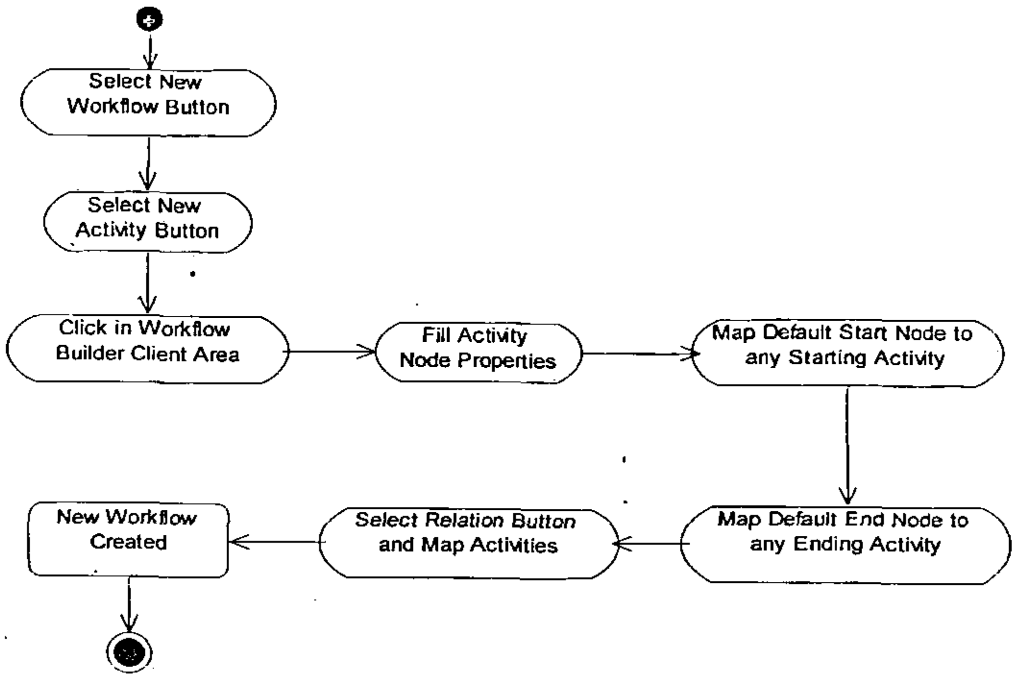

Select New Workflow Button
Select New Activity Button
Click in Workflow Builder Client Area
Fill Activity Node Properties
Map Default Start Node to any Starting Activity
Map Default End Node to any Ending Activity
Select Relation Button and Map Activities
New Workflow Created

Fig. 4.2. Create Workflow

### 4.1.3 Load Workflow

The following steps will be accomplished whenever user saves the Workflow in the database.

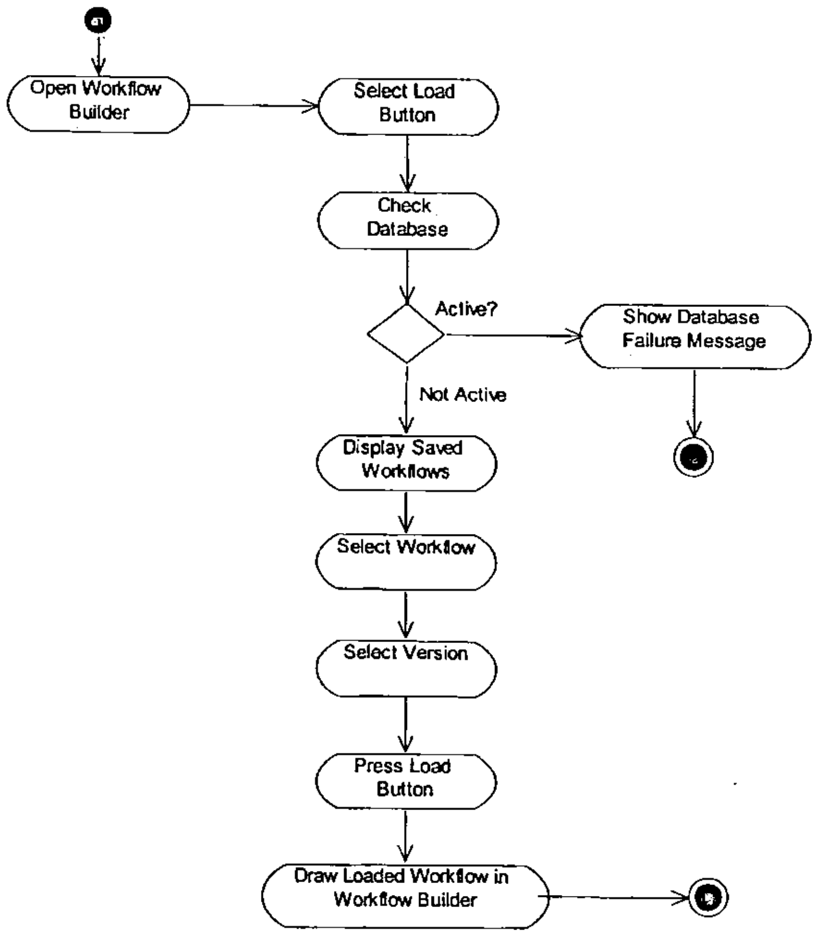


Fig. 4.3 Load Workflow

## 4.2 Interaction Diagrams

An interaction diagram shows one of the flows through a use case. Every use case correspondingly has an interaction diagram. Integration diagram contains:

- *Object: An Interaction diagram can use object names, class name or both.*
- *Messages:* Through a message, one object or class can request that another carry out some specific function. For example, a form may ask a report object to print itself.

There are two types of interaction diagrams:

- Sequence Diagram – ordered by time
- Collaboration Diagram – not ordered by time.

### 4.2.1 Sequence Diagram

Sequence diagrams are ordered by time. They are useful if some one wants to review the flow of logic through a scenario. Although Collaboration diagrams include sequencing information, it is easier to see on a Sequence diagram.

Before describing the sequence diagram of the miscellanea, we would like to describe the server side sequence diagram which will be used in almost all of the next coming sequence diagrams of the Miscellanea.

In the following sequence diagram we will describe the creation and removal of the bean managed persistence. This scenario appears when the bean managed entity bean handles client request.

### 4.2.2 Collaboration Diagram

Collaboration diagrams are useful if we want to assess the impact of a change. It is easy to see on a collaboration diagram, which objects communicate with which other objects. If we need to change an object, we can easily see which other objects might be affected. A collaboration diagram shows the same information, but is organized differently. Sequence diagram can show focus of control; Collaboration diagrams can show a data flow.

## 4.3 Interaction Diagrams of DPSFLOW

Interaction diagrams of DPSFLOW are shown in the following pages. These contain the sequence diagrams. Since collaboration diagrams help us in impact of change therefore they are not as much important in our project. Sequence diagrams show completely the flow of logic.

### 4.3.1 Initialize Workflow Builder

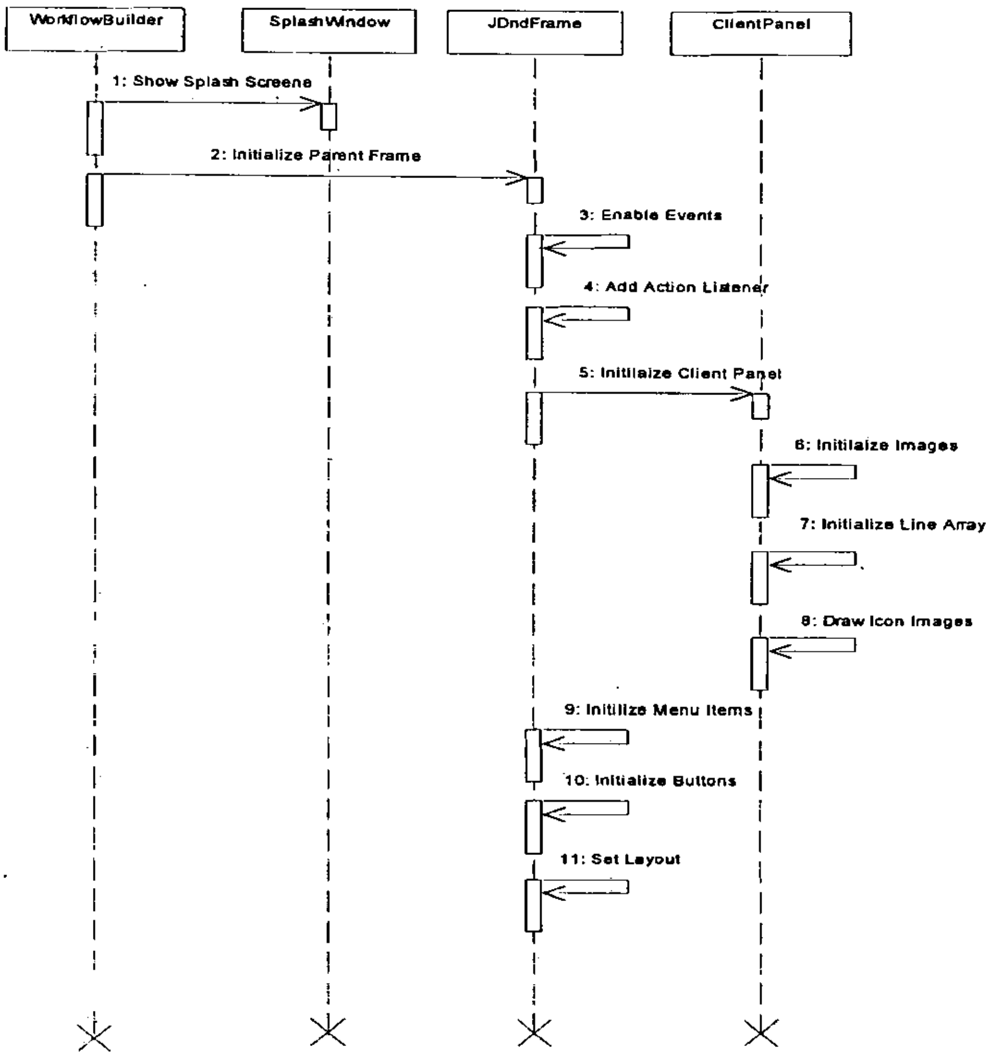


Fig. 4.4 Initialize Workflow Builder

### 4.3.2 Create Workflow

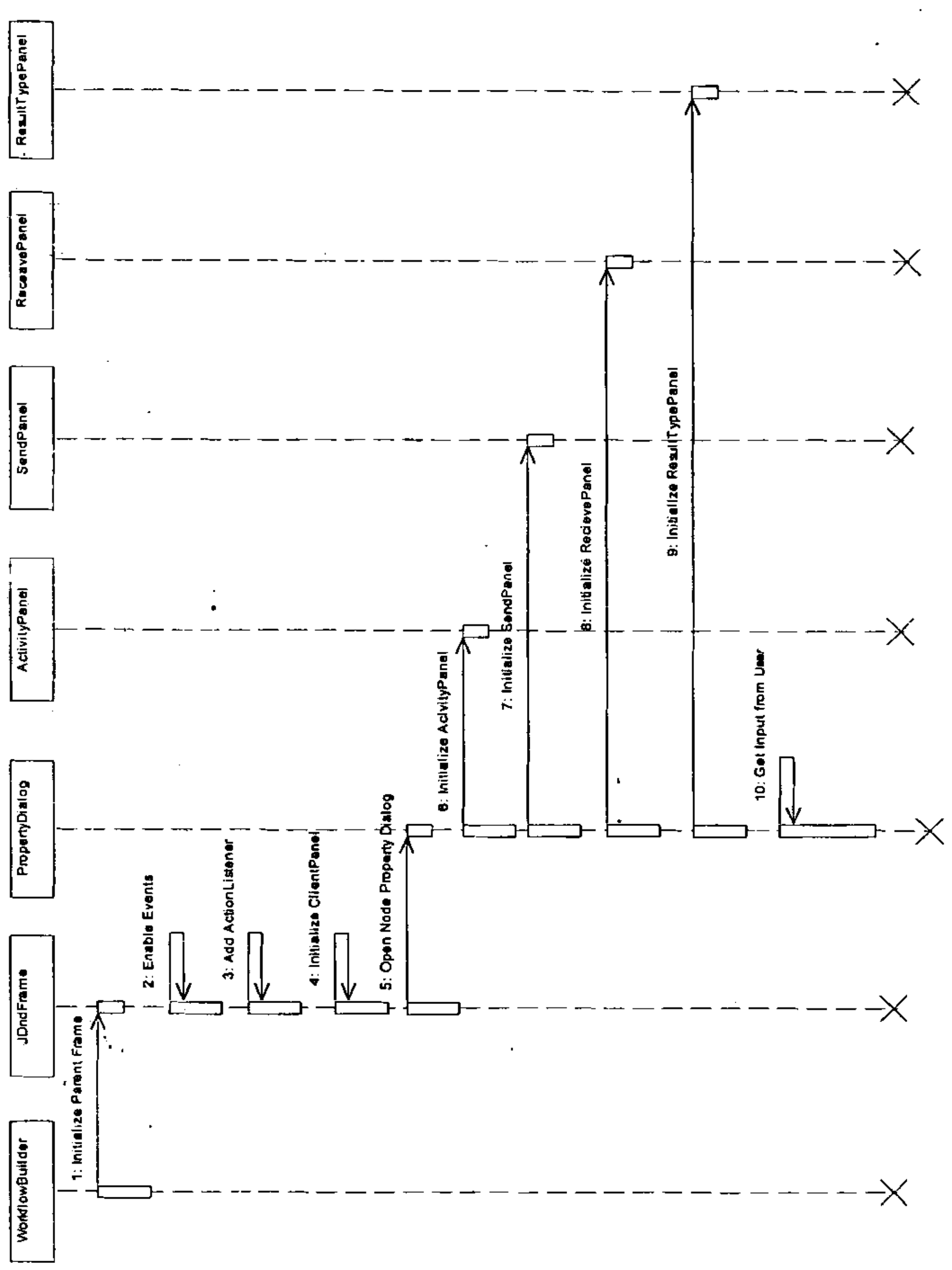


Fig. 4.5 Create Workflow

## 4.3.3 Save Workflow

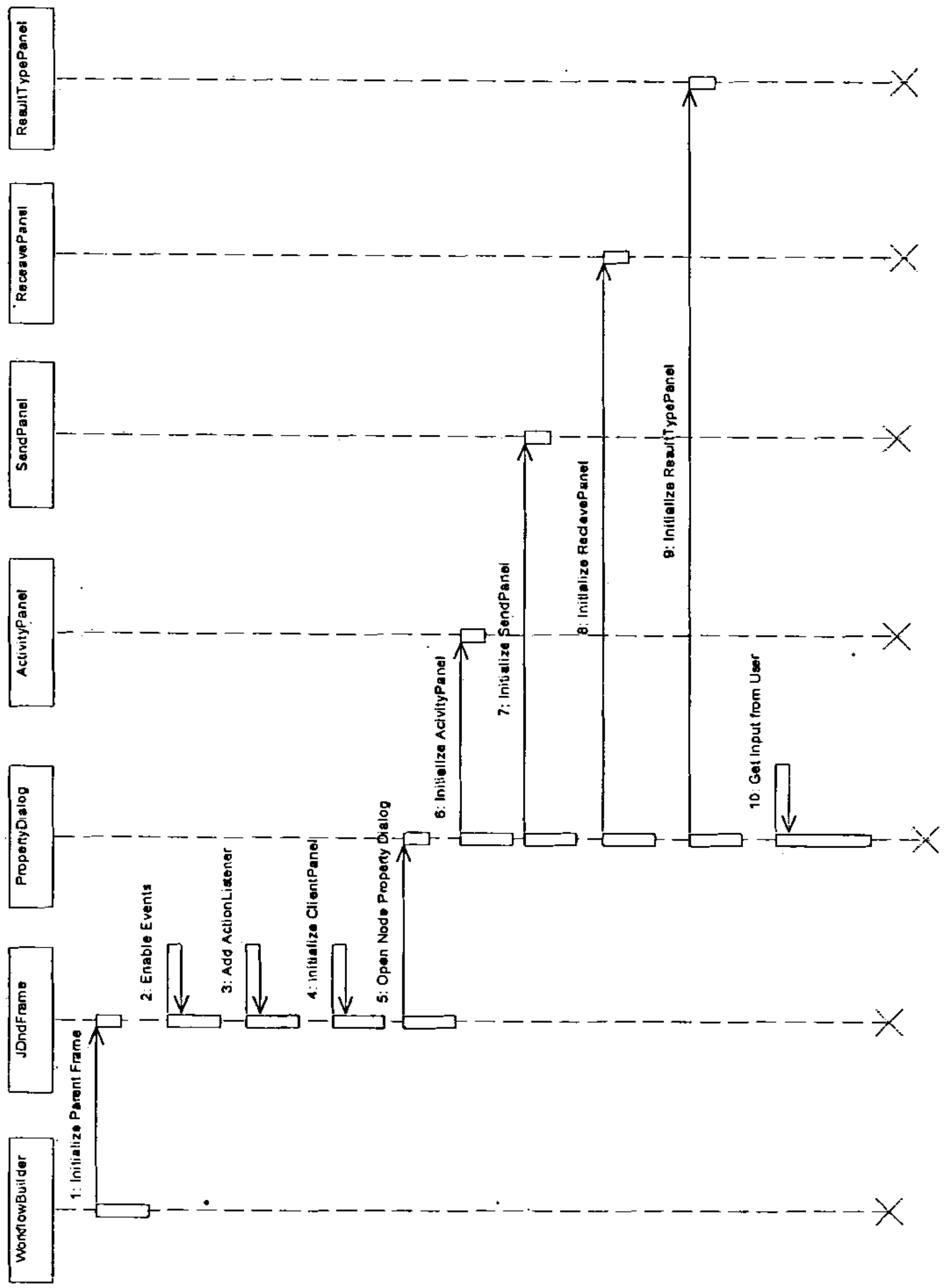


Fig. 4.6 Save Workflow

### 4.3.4 Load Workflow

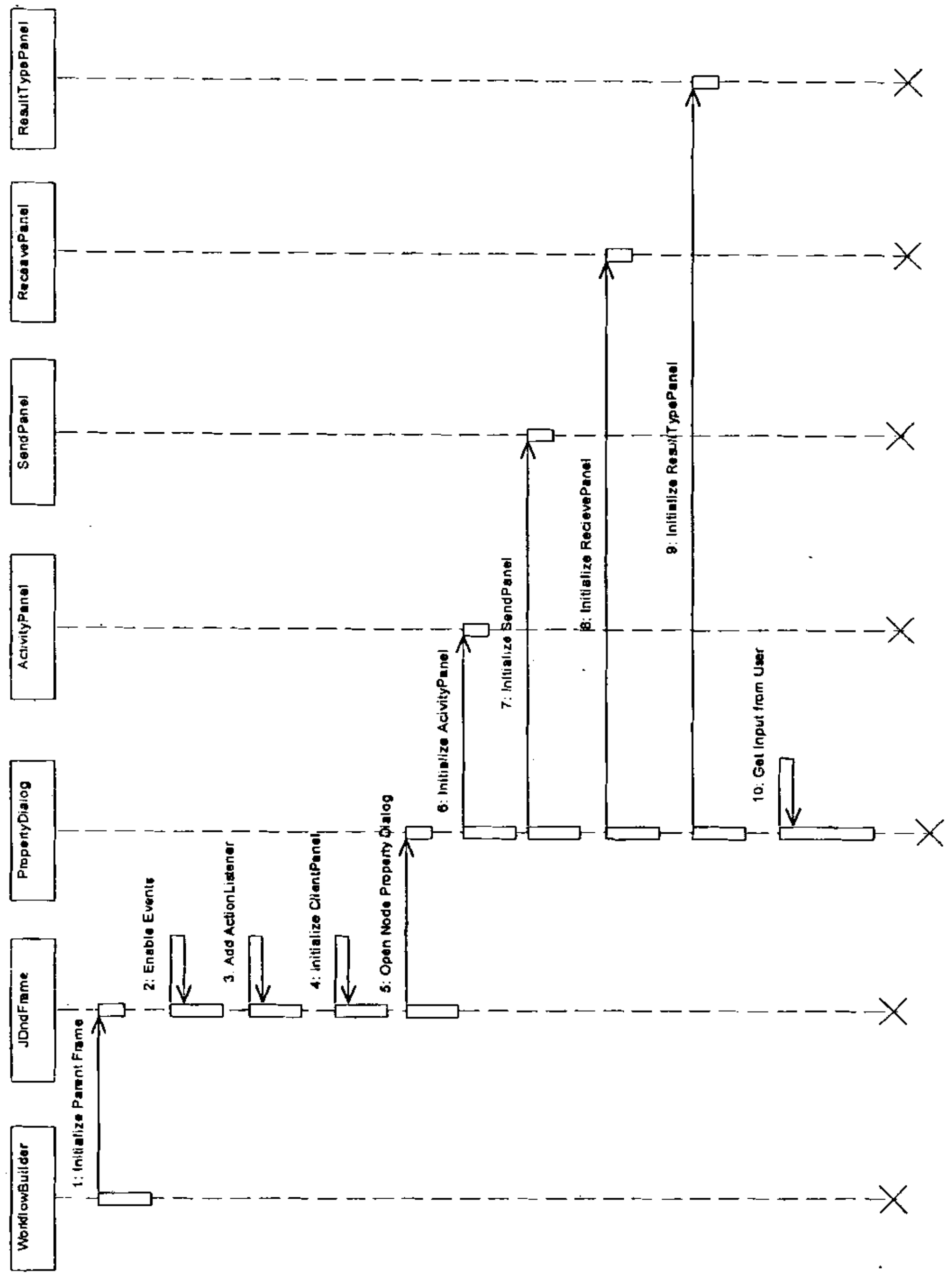


Fig. 4.7 Load Workflow

## 4.4 Conceptual Model

The result of the analysis task is the conceptual model of the problem domain defined by classes relevant to the domain and associations between these classes. The goal is to capture the domain semantics with as little concern as possible of the navigational and presentational aspects. Activities of the conceptual design step are to find classes, to specify attributes and operations, to define hierarchical structures and to determine sub-systems. Well-known object-oriented modeling techniques, such as aggregation, composition, generalization and specialization are used to achieve this purpose. Classes and associations are described by attributes and operations; they can be organized into UML packages. The conceptual model is the starting point for the design of one or more navigational design, i.e. more than one hypermedia application. It is represented by a UML class diagram.

## 4.5 Navigational Class Model

The navigational class model defines a view on the conceptual model showing which classes of the conceptual model can be visited through navigation in the application. This model is built with a set of navigational classes and associations, which are obtained from the conceptual model, i.e. each navigational class and each association of the navigational class model is mapped to a class, respectively to an association in the conceptual model. In the navigational class model navigability is specified for associations, i.e. direction of the navigation along the association is indicated by an arrow attached to the end of the association's line. It is possible to attach an arrow to both ends of an association. In this case the user can navigate in both directions along the association. We distinguish for each link a navigational source object and a navigational target object; the latter one may be determined dynamically.

A navigational class is defined as a stereotyped class << navigational class >> with the same name as the corresponding class of the conceptual model. Navigational objects are instances of these navigational classes connected by links (in UML terms) that are instances of the associations of the navigational model. The navigational class model can be seen as a sub-graph of the conceptual model where classes which are not needed for navigation are eliminated or reduced to attributes of other classes. The values of these attributes can be computed from some conceptual objects. Navigational classes and associations are graphically represented in a UML class diagram.

## 4.6 DPSFLOW Navigational Class Model

Following is the Navigational Class Model of DPSFLOW Management System.

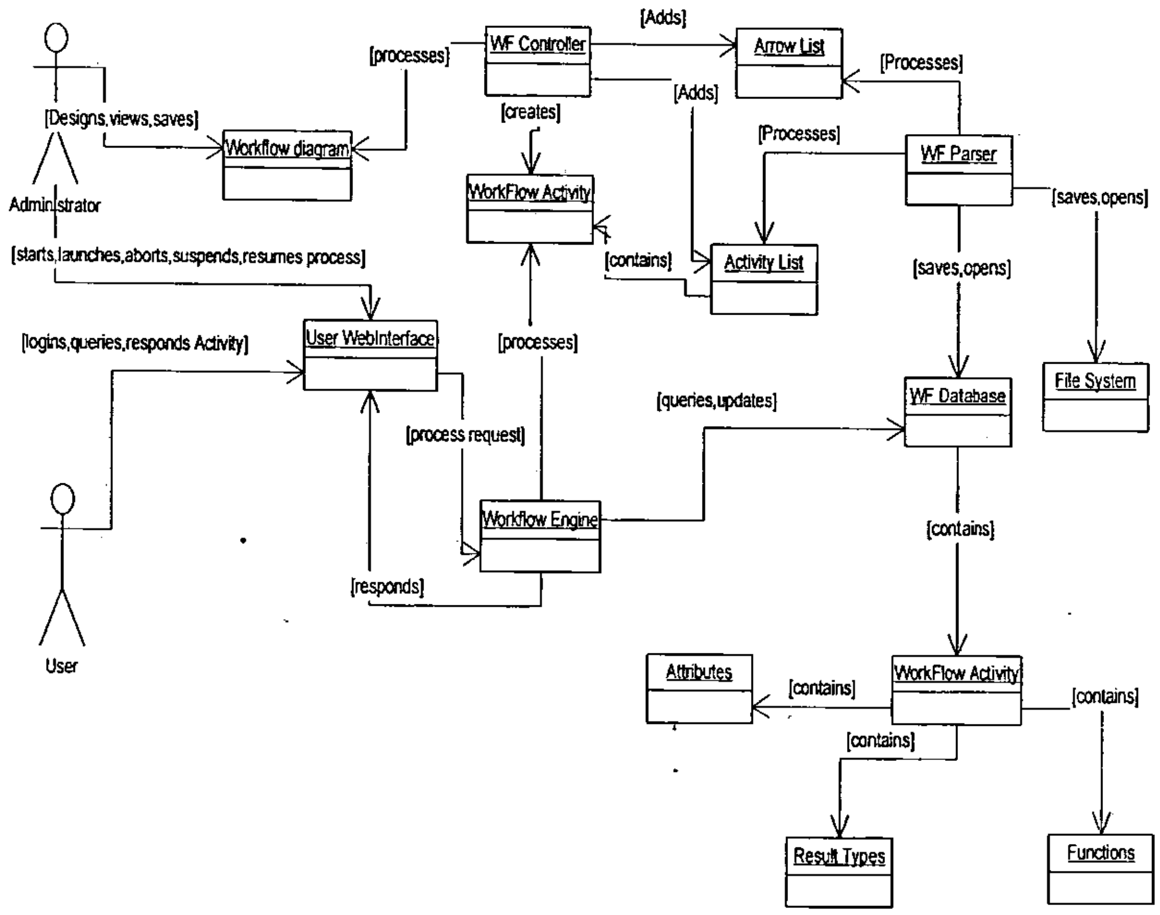


Fig. 4.8 Navigational Class Model

## 4.7 Class Diagrams

A class diagram is similar to a family tree. A class diagram consists of a group of classes and interfaces reflecting important entities of the business domain of the system being modeled, and the relationships between these classes and interfaces. The classes and interfaces in the diagram represent the members of a family tree and the relationships between the classes are analogous to relationships between members in a family tree. Interestingly, classes in a class diagram are interconnected in a hierarchical fashion, like a set of parent classes (the grand patriarch or matriarch of the family, as the case may be) and related child classes under the parent classes.

Similarly, a software application is comprised of classes and a diagram depicting the relationship between each of these classes would be the class diagram.
*By definition, a class diagram is a diagram showing a collection of classes and interfaces, along with the collaborations and relationships among classes and interfaces.*

A class diagram is a pictorial representation of the detailed system design. Design experts who understand the rules of modeling and designing systems design the system's class diagrams. A thing to remember is that a class diagram is a static view of a system. The structure of a system is represented using class diagrams. Class diagrams are referenced time and again by the developers while implementing the system.

### 4.7.1 Elements of a Class Diagram

A class diagram is composed primarily of the following elements that represent the system's business entities:

#### 4.7.1.1 Class

A class represents an entity of a given system that provides an encapsulated implementation of certain functionality of a given entity. These are exposed by the class to other classes as *methods*. Apart from business functionality, a class also has properties that reflect unique features of a class. The properties of a class are called *attributes*.

### 4.7.1.2 Interface

An interface is a variation of a class. As we saw from the previous point, a class provides an encapsulated implementation of certain business functionality of a system. An interface on the *other hand provides only a definition of business functionality of a system. A separate class implements the actual business functionality.*

### 4.7.1.3 Package

A package provides the ability to group together classes and/or interfaces that are either similar in nature or related. Grouping these design elements in a package element provides for better readability of class diagrams, especially complex class diagrams.

## 4.8 DPSFLOW Class Diagrams

In DPSFLOW we have made following Classes that were created when we generalized our Navigational Class Model.

### 4.8.1 WfConnection

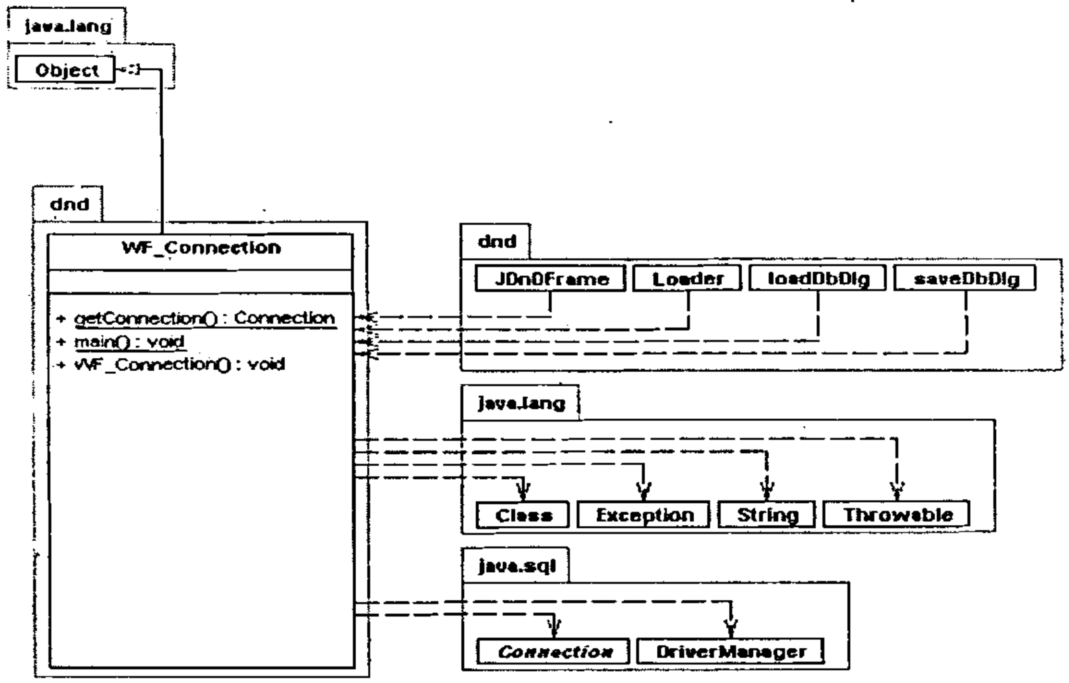


Fig. 4.9 Class Diagram of WFConection

### 4.8.2 Activity

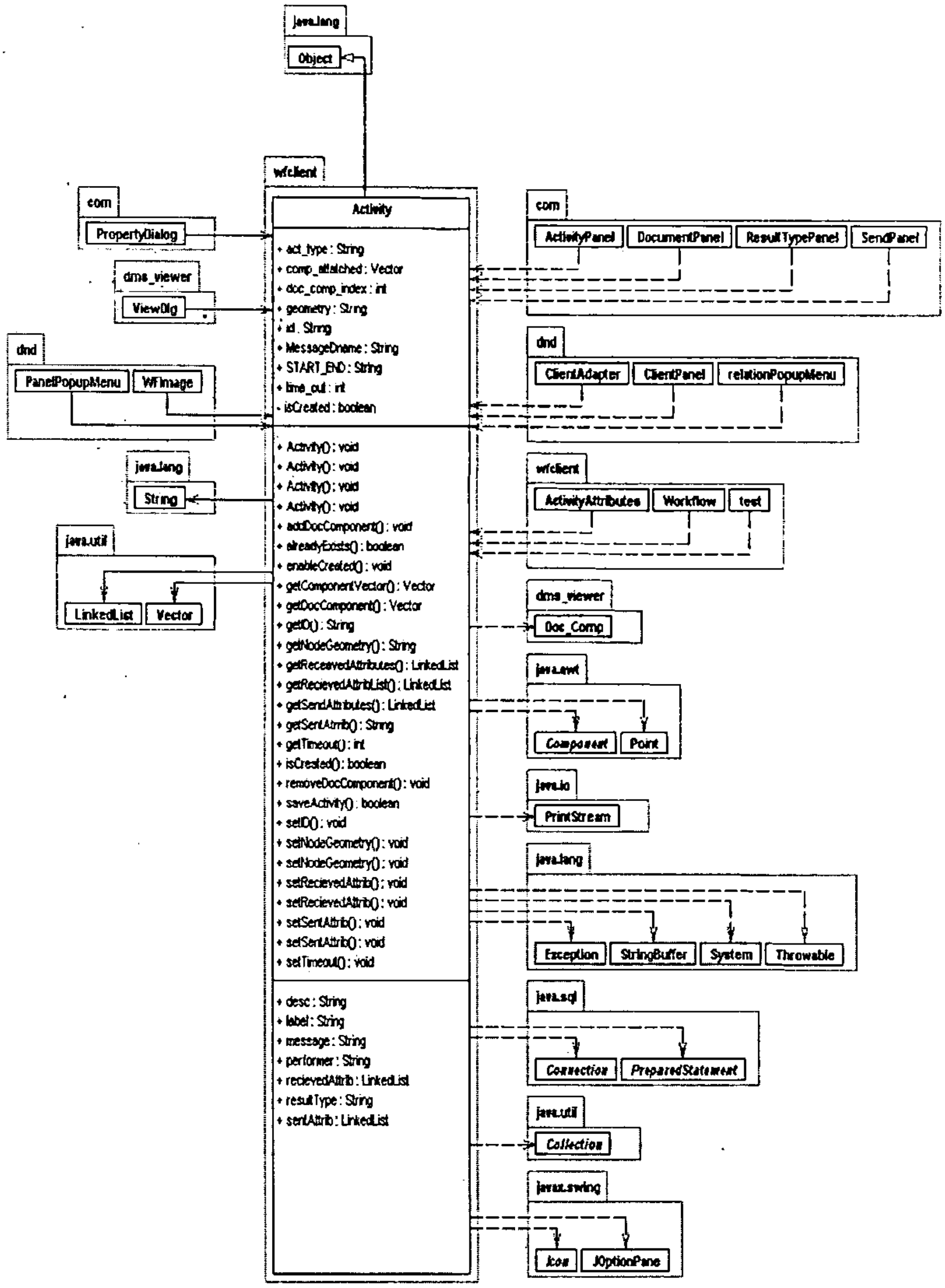


Fig. 4.10 Class Diagram of Activity

## 4.8.3 ActivityAttributes:

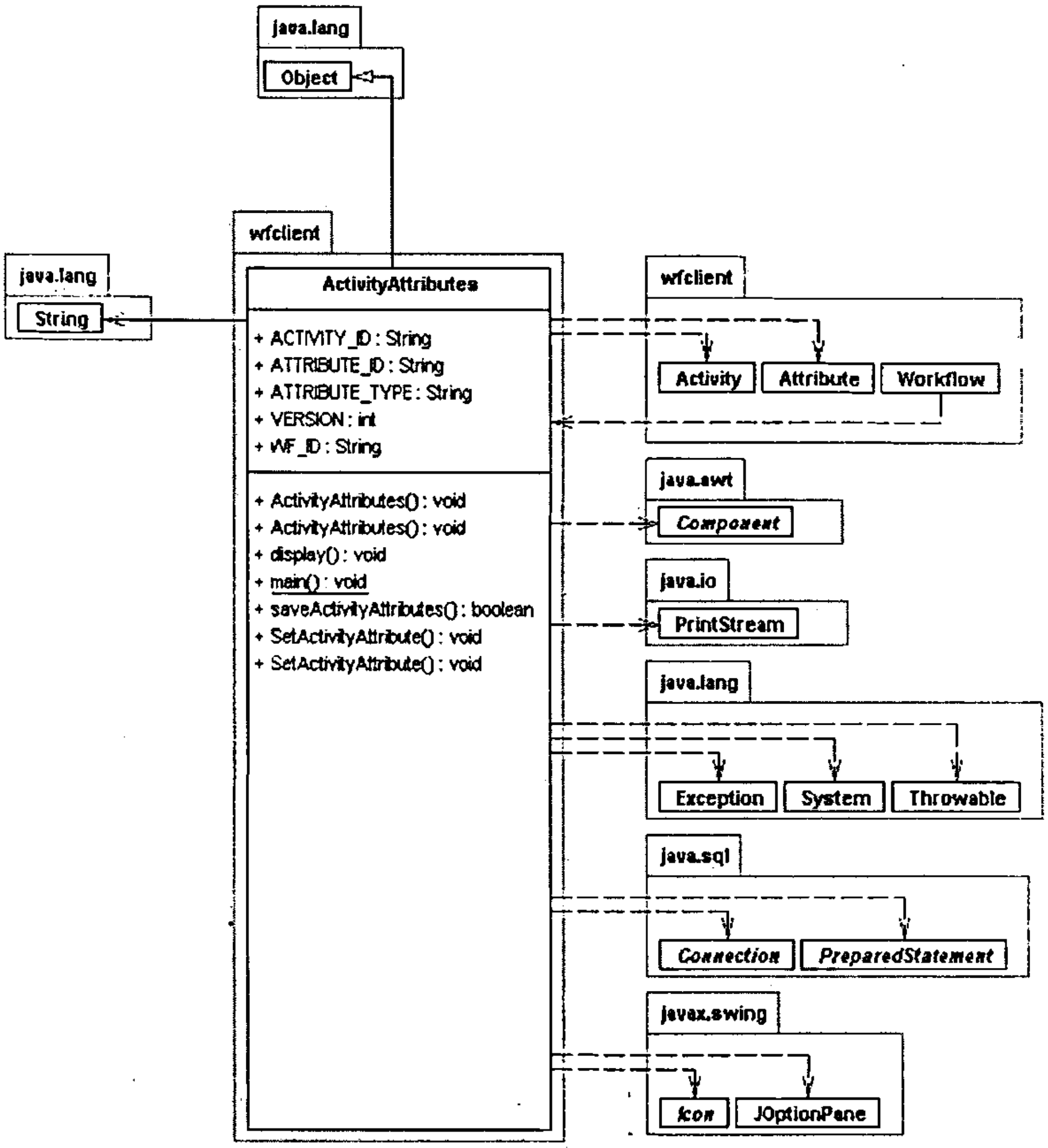


Fig. 4.11 Class Diagram of ActivityAttributes

### 4.8.4 Attribute:

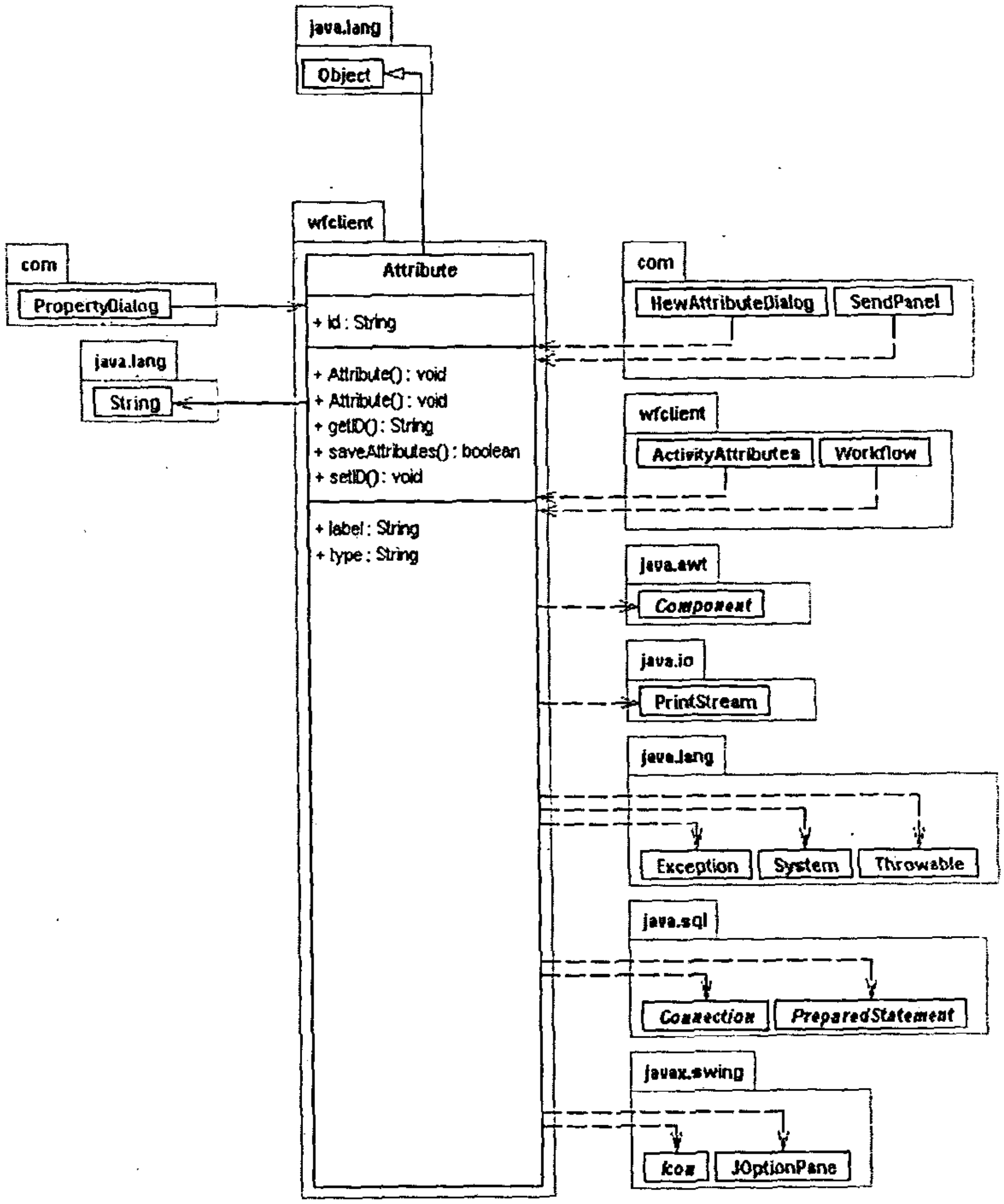


Fig. 4.12 Class Diagram of Attribute

## 4.8.5 ProcessActivities

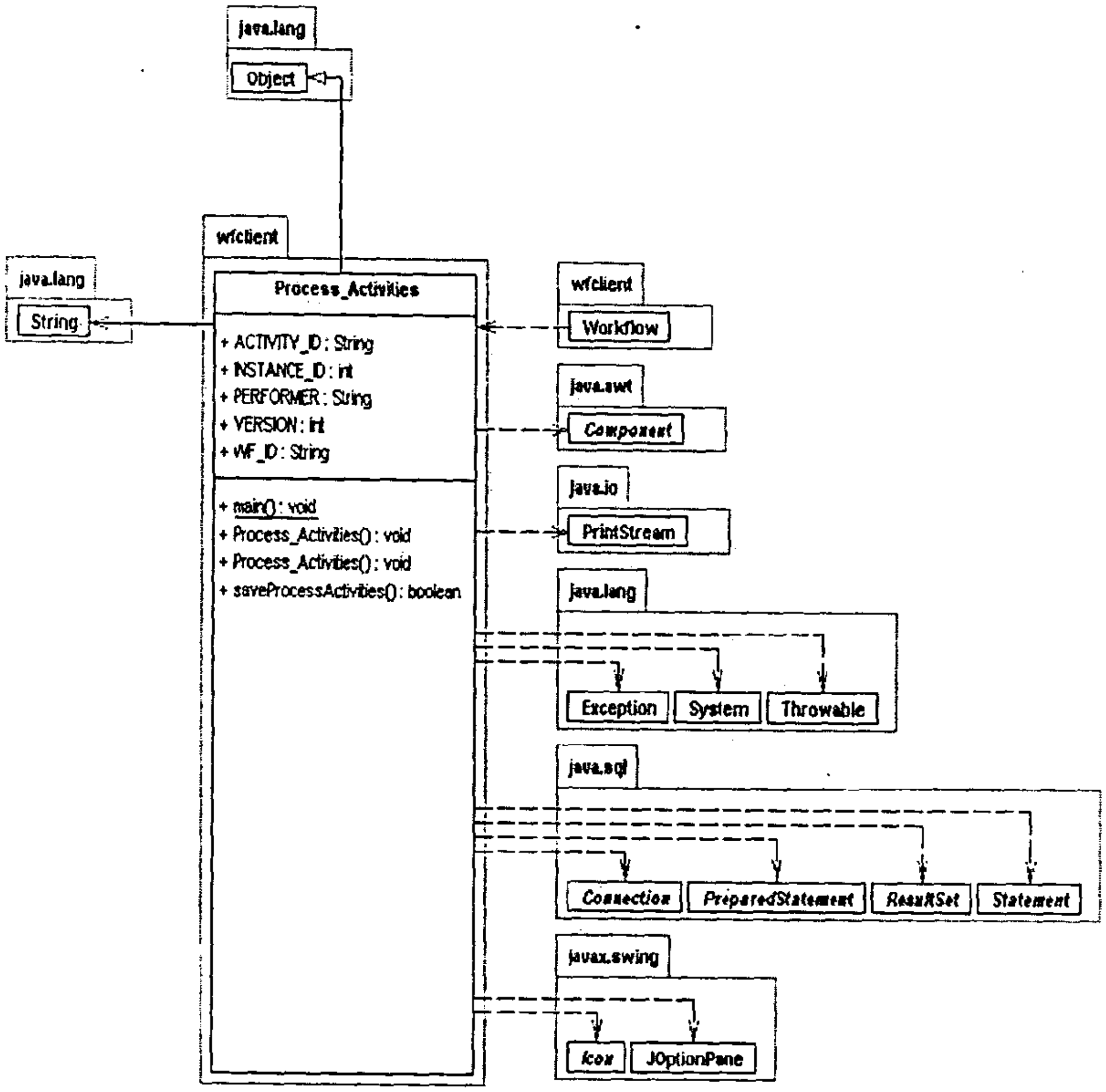


Fig. 4.13 Class Diagram of ProcessActivities

## 4.8.6 Relation

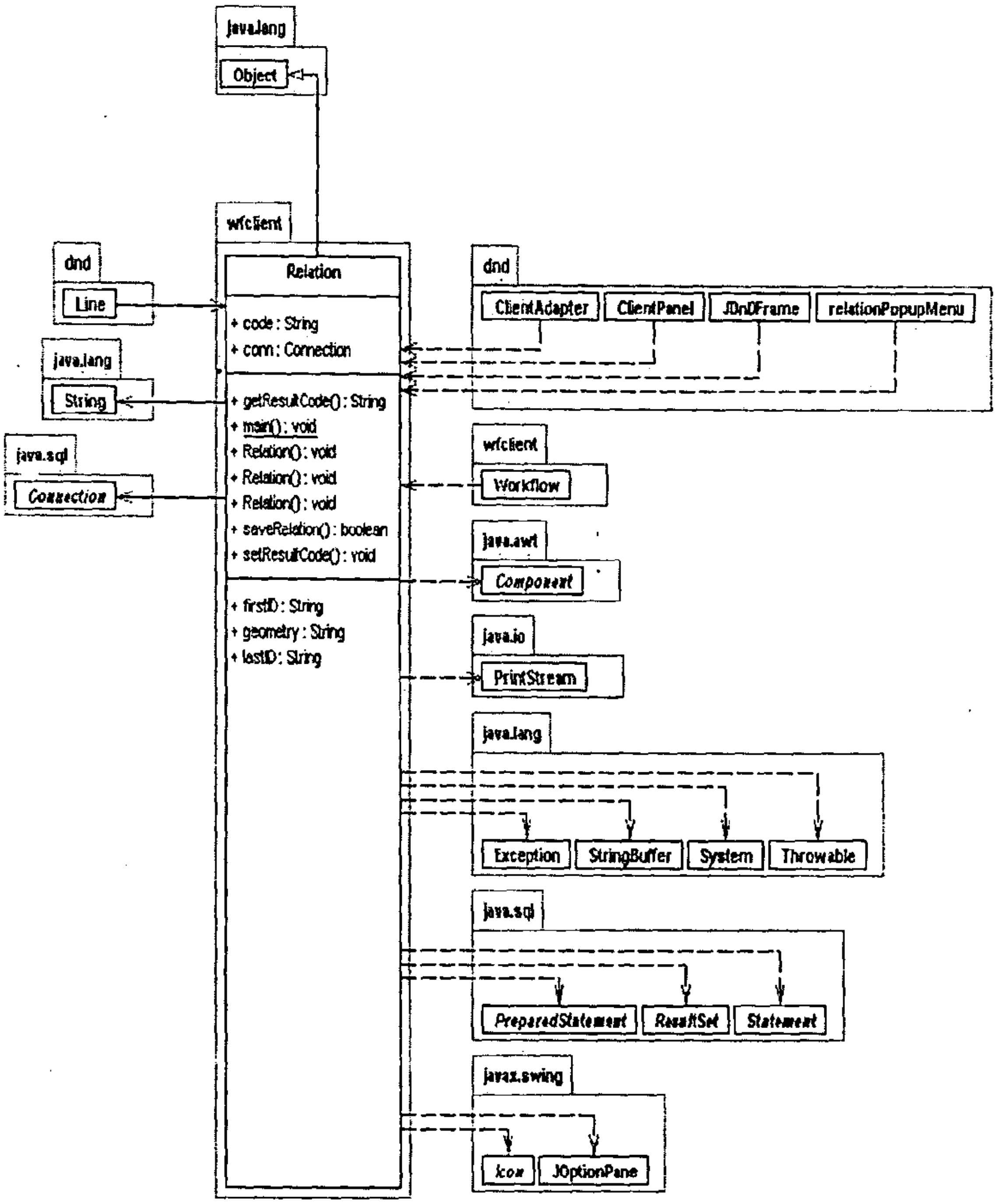


Fig. 4.14 Class Diagram of Relation

## 4.8.7 ResultType

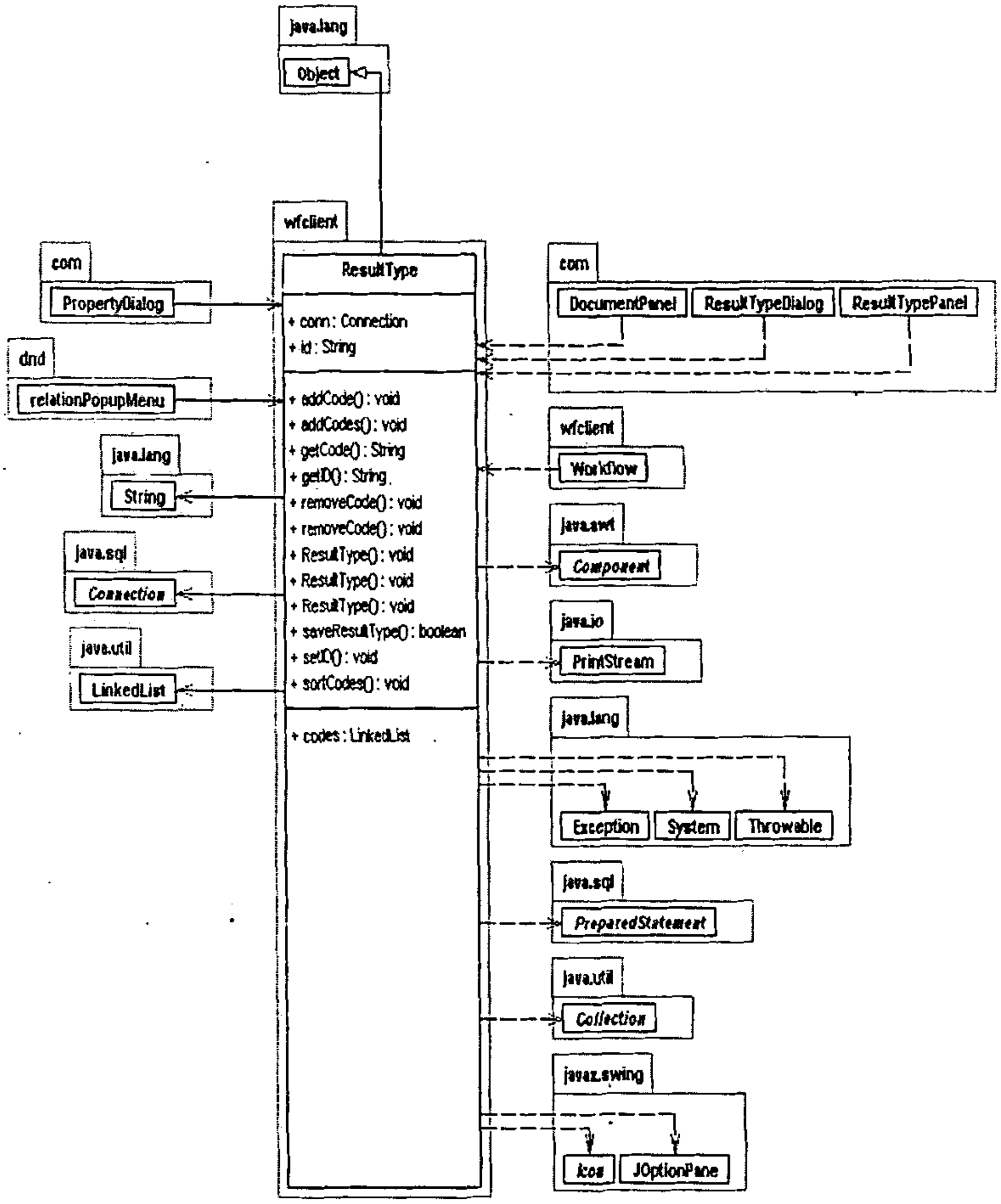


Fig. 4.15 Class Diagram of ResultType

### 4.8.8 SaveDbDlg

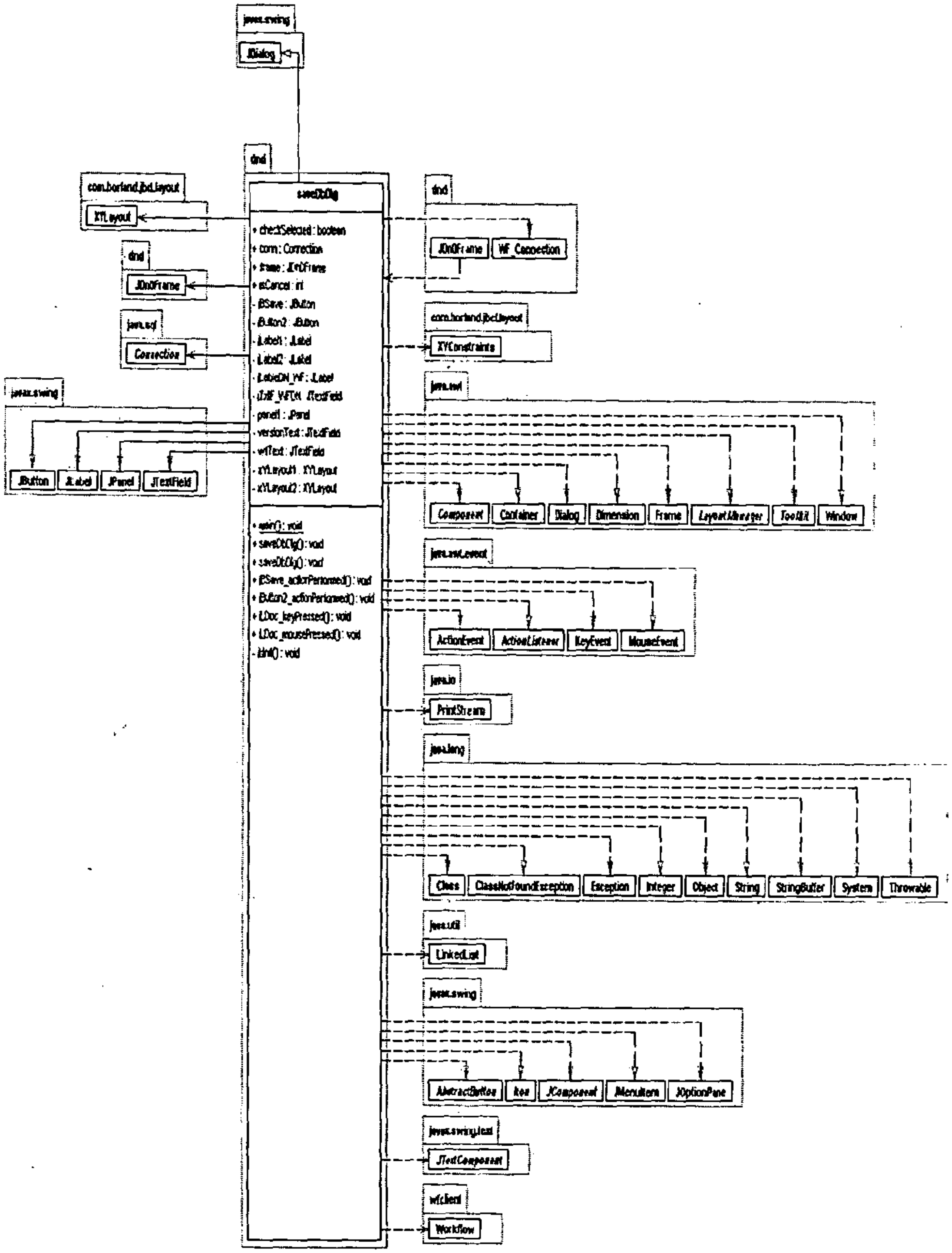


Fig. 4.16 Class Diagram SaveDbDlg

## 4.8.9 SplashScreene

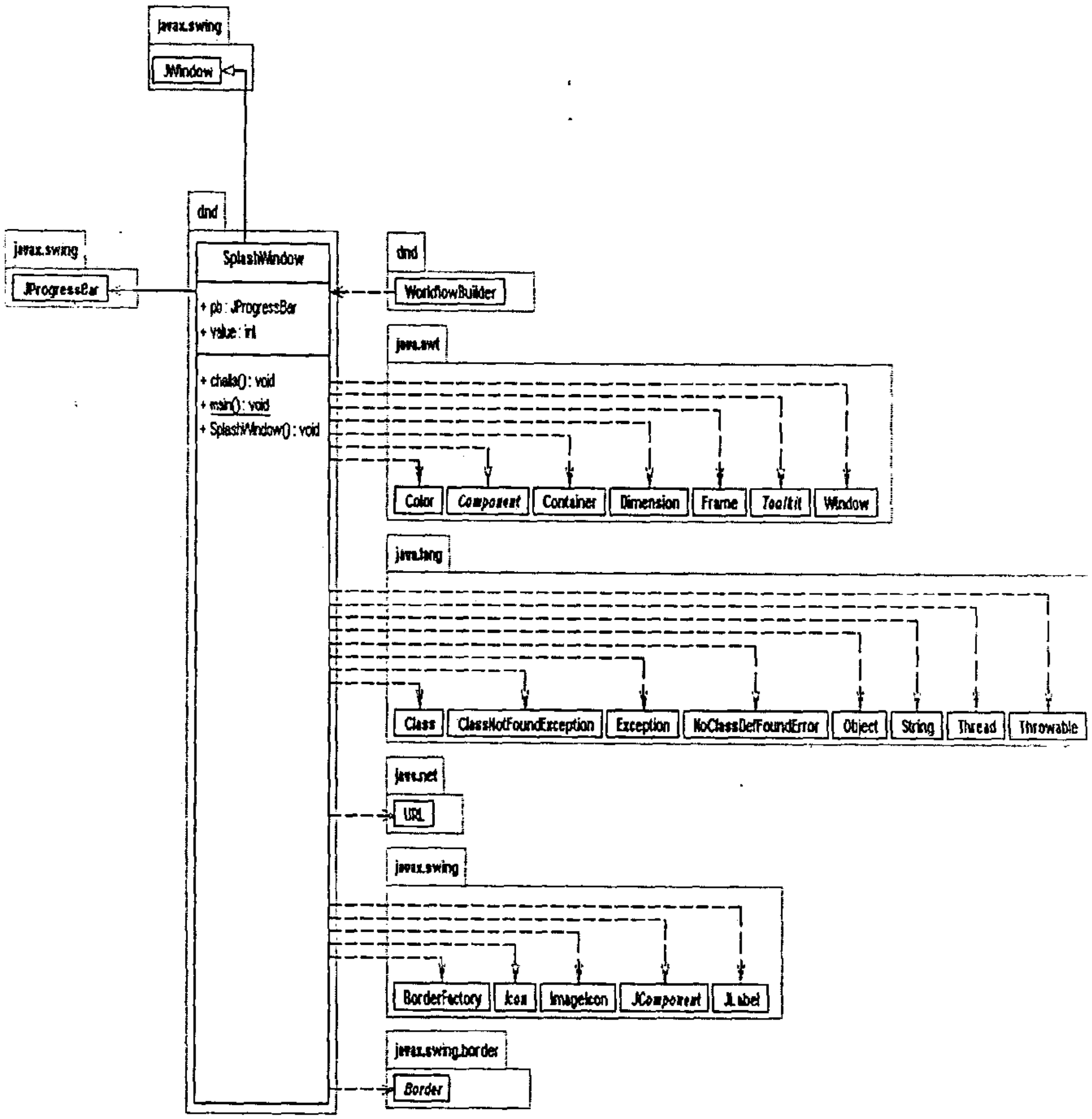


Fig. 4.17 Class Diagram of SplashScreene

## 4.8.10 WorkFlowBuilder

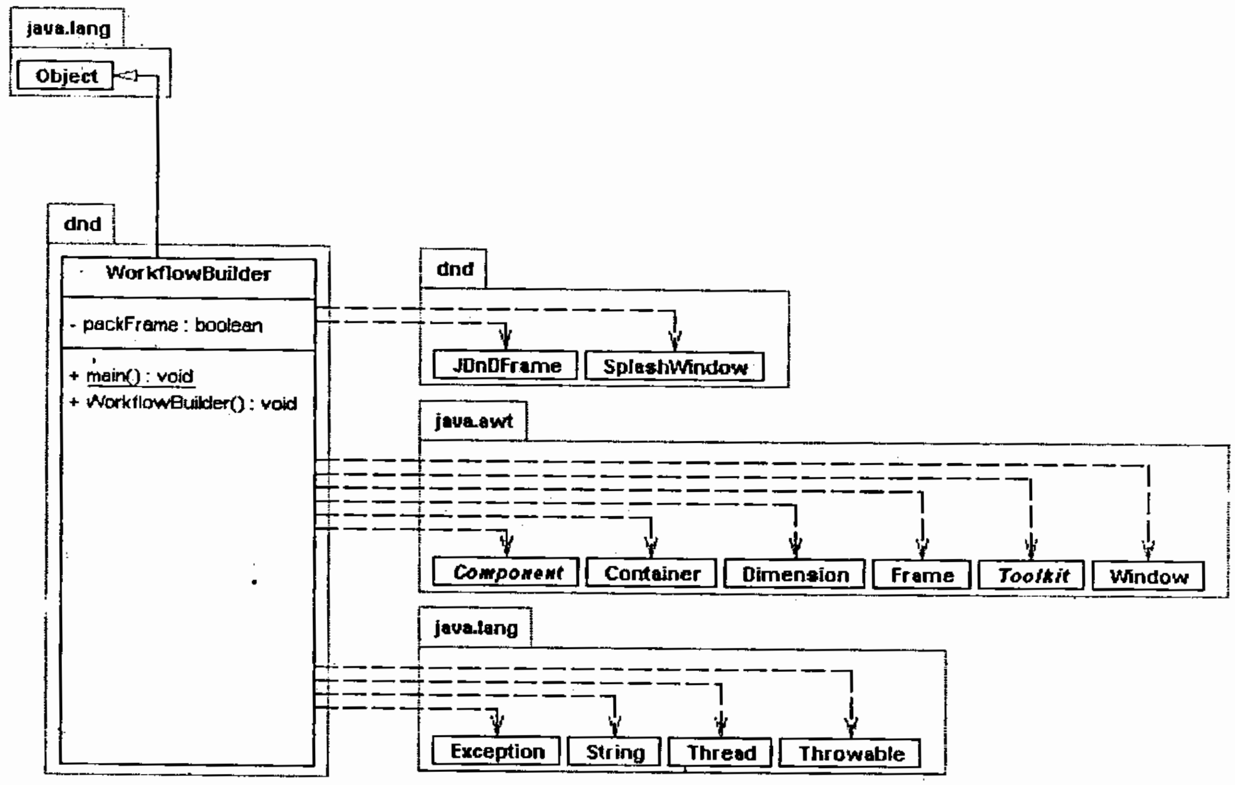


Fig. 4.18 Class Diagram of Workflow Builder

## 4.8.11 WFClient

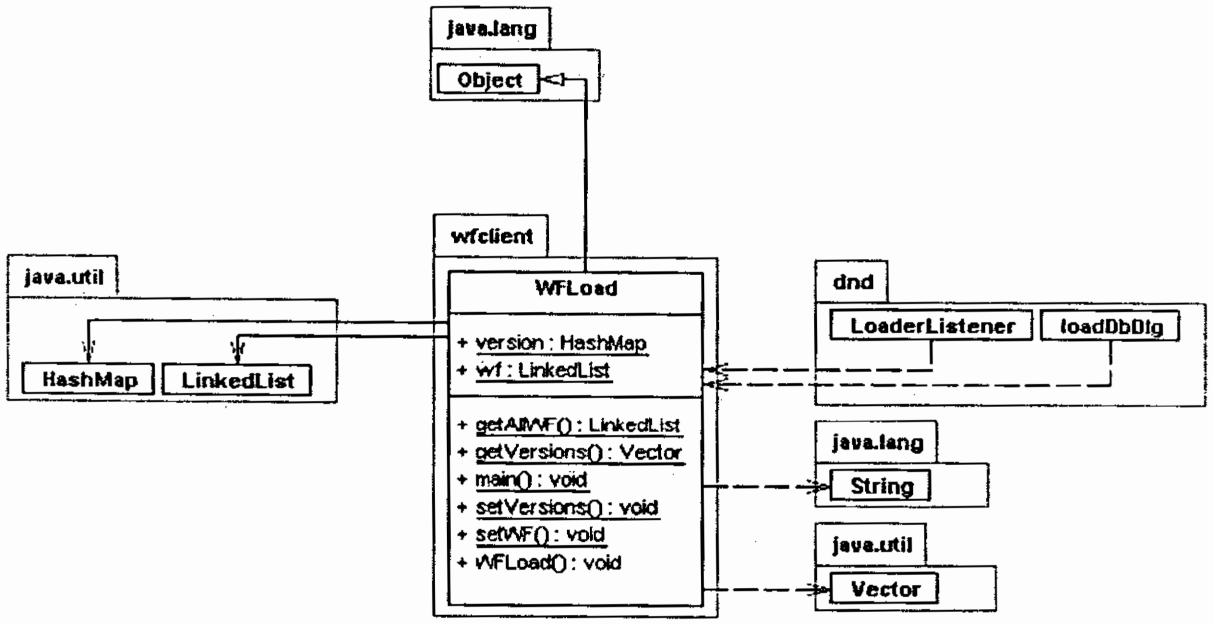


Fig. 4.19 Class Diagram of WFClient

## 4.8.12 WorkflowProperties

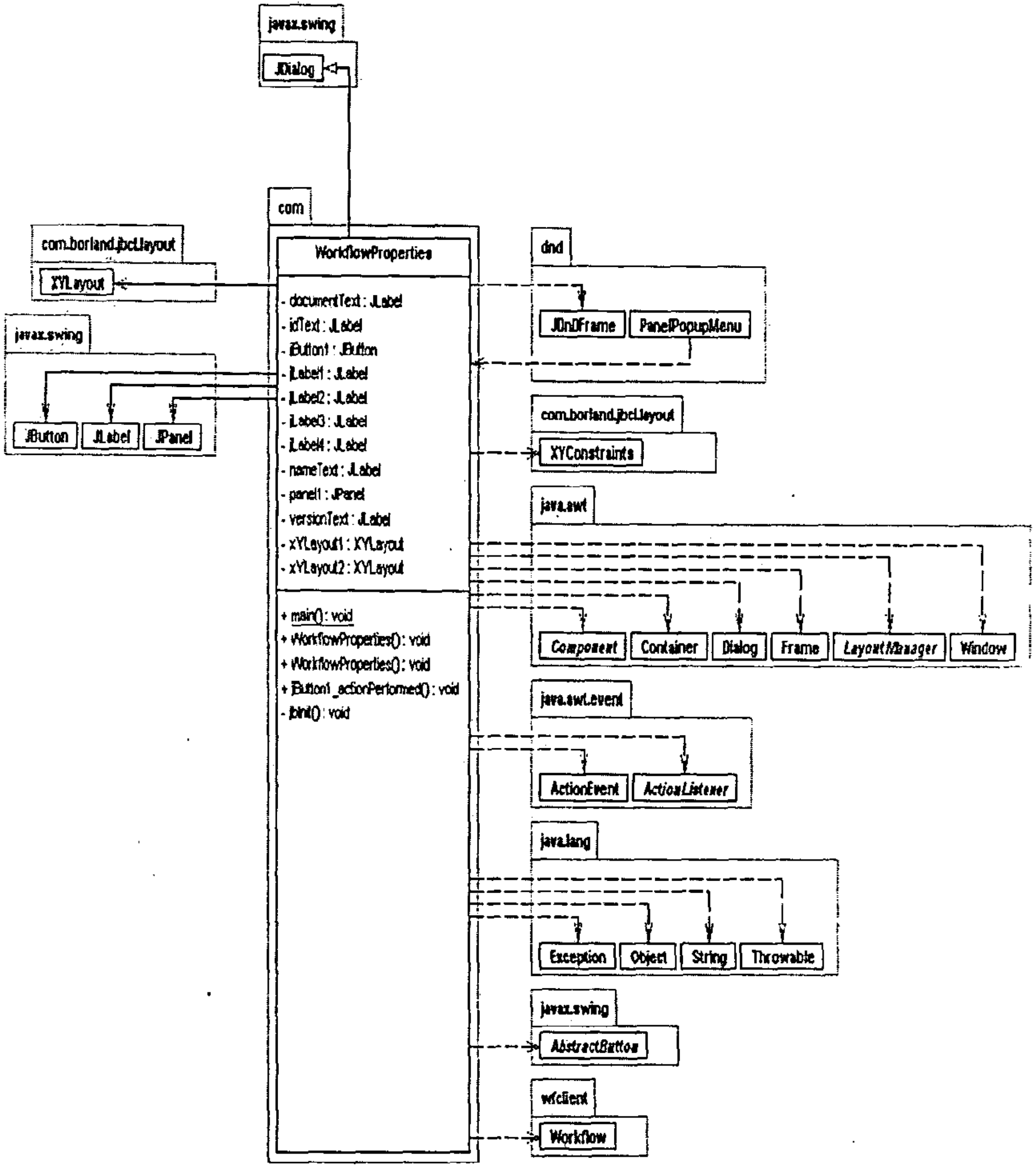


Fig. 4.20 Class Diagram of WorkflowProperties

## 4.8.13 Workflow

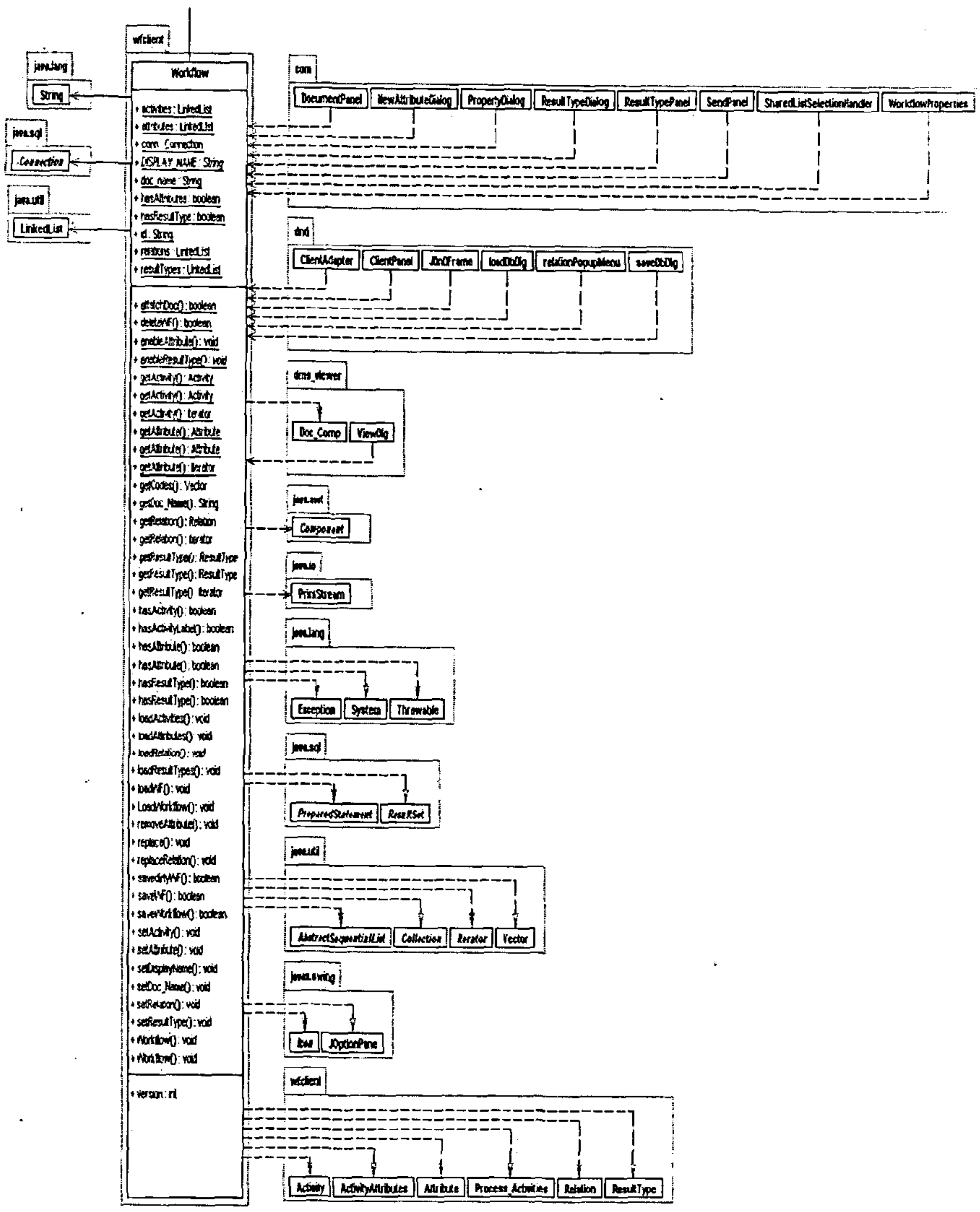


Fig. 4.21 Class Diagram of Workflow

# Chapter 5

# DATABASE ARCHITECTURE & DESIGN

# 5. Database Architecture & Design

The database needs a structure definition to be able to store data and to recognize the content and be able to retrieve information. The structure has to be developed for the need of applications, which help us perform a business process to achieve an added value for the customer.

Database implementation is the final step in developing a business supporting application. It has to comply with the requirements of the business process, which is the first abstraction of the view of the database.

## 5.1 Logical Database Design

The process of constructing a model of the information used in an enterprise based on a specific data model, but independent of a particular DBMS and other physical considerations.

The decision on the kind of data storage is the step from the conceptual to logical database design, as the logical database design describes a DBMS related representation, and the conceptual design the target independent process including the logic of the application.

This decision is not always easy when talking about long-term transient data, which is used just until the application stops and not used very often. Examples for long-term transient data are web-server component data, like transaction data, user data, etc.

We would like to expand the definition and include the granularity and language of description. With the logical database design the natural language and granularity of objects has to be used. Any differentiation from the base element would influence the result of the design and focus already on a specific architecture.

The process of constructing a model of information used in an enterprise based on a specific data model, using natural objects of information and natural associations between them. The model of information is independent of a particular implementation and other physical consideration.

The logical database design is the model of valuable information. It describes the information itself and the relationships between information. The kind of relationships is not limited, because any limitation would affect the structure of information.

The description of the information is implementation neutral. It uses abstraction to depersonalize objects from the conceptual database design. Objects of the same kind are grouped together to one persistent class. The objects are still at the same granularity level as the global requirements.

Persistent classes contain attributes, which represent the basic element of persistent storage. An attribute can store any kind of data independent of the later physical limitation. DBMS independent attribute types are used on the logical level to prevent any physical influence.

The accessibility of the persistent data is defined by operations on a class. In the design process operations evolve from messages at interaction diagrams. Operations can be public or private, in which case they deal with the internal integrity of the information. Private operations are mostly defined out of the complexity of the business process, which is best recognized in the use case realization. They are rules for complex relations between information.

The association between classes can represent any kind of influence. The association does not have any influence on the attributes, and the attributes do not have any influence on the association.

Since the logical model does not consider any particular implementation, the associations may include aggregation (often called part of association), inheritance (type of association), and any kind of cardinality (including m to n associations).

Following is the detailed logical view of our database design.

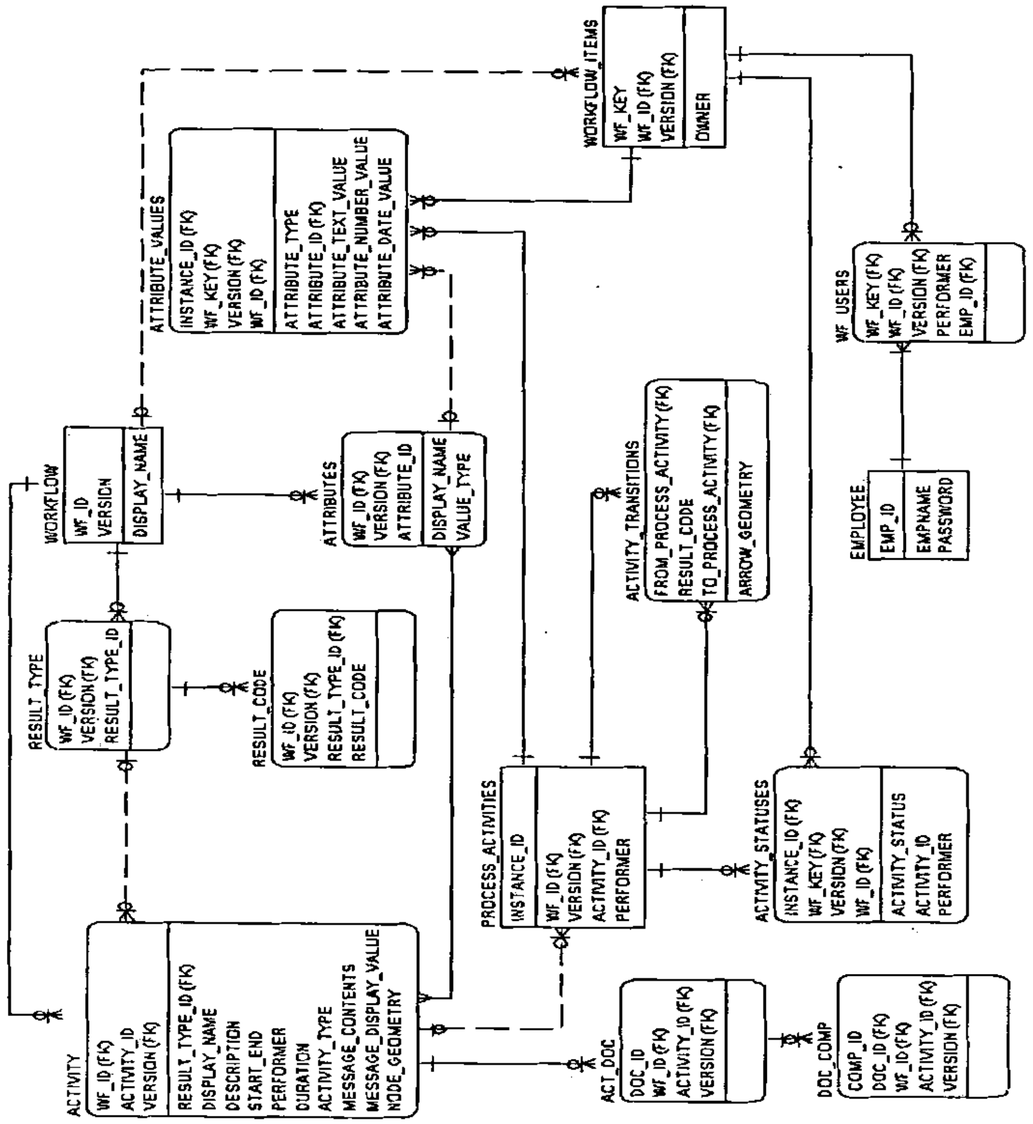


Fig. 5.1 Logical Database Design of DPSFLOW

The logical design is most commonly expressed in a class diagram, which is a partial view on the whole design. The resulting artifacts of the logical design are persistent classes and the associations between classes, as well as the essential constraints on the data. The logical database design handles the information view on the enterprise. It contains most important information in the enterprise as well as the associations between information. As a difference to the conceptual database design, logical database design does not contain the process itself, because the process is further evolved in the application design.

## 5.2 Physical Database Design

Physical Database Design is process of producing a description of the implementation of the database on secondary storage. It describes the storage structures and access methods used to achieve efficient access to the data.

The physical database design defines the way to implement the data storage. It is target dependent and complies with the storage technology, the instance of the database, the architecture, and the implementation rules.

The physical database design has to remodel the artifacts of the logical design according to the used storage technology, which can be hierarchical, network, relational, object-relational, object-oriented, or multidimensional.

The aim of the physical design is for the relational database to define the tables and detailed constraints for the in the logical database design specified data model. The columns of a table are specified with the details on internal representation.

Associations are translated to relationships, which are represented by key constraints. Operations can be implemented as all kinds of access methods. Together they form information out of pure data.

An important view on data is the security, which can be easily defined out of the use case realizations and the logical database artifacts. It is important to have the global overview of the application and data storage to be able to design security schemas.

The resulting artifact of the physical database design is the exact description of the implementation. Figure on the next page is the database design of our model.

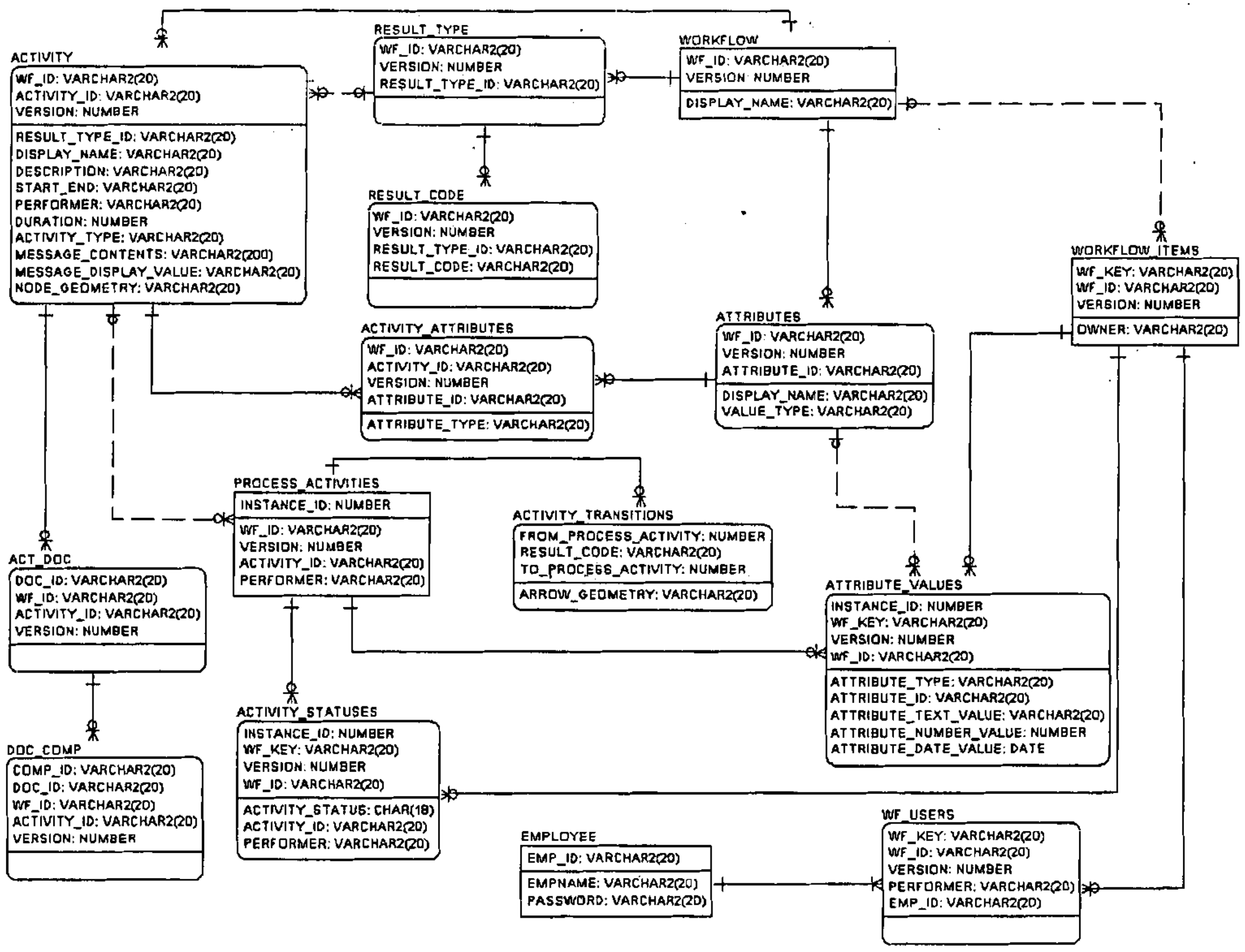


Fig. 5.2. Physical Database Design of DPSFLOW

## 5.3 Normalization of Tables

Normalization is the process of grouping data elements according to specific "normalization" rules in order to minimize processing difficulties. Normalization is usually done in three steps, or forms (i.e. first normal form, etc.), but some have extended this to as many as five steps (fifth normal form).

Normalization of table means to reduce anomalies, occur in the tables, so that the database design becomes efficient and good. A good design principle is "*one fact in one place*" i.e. avoid redundancy in tables. Normalization exist around the concept of normal forms. A relation is said to be in a particular normal form if it satisfies a certain prescribed set of conditions. For example, a relation is said to be in first normal form (abbreviated 1NF) if and only if tit satisfies the condition that it contains scalar values only.

### 5.3.1 First Normal Form

A relation is said to be in first normal form if and only if all underline domains contain atomic values only.If we see our relations, we can say that, there is no attribute to contain non-atomic entry.

### 5.3.2 Second Normal Form

A relation that is in first normal form and has every nonkey attribute fully functionally dependent on the primary key is called second normal form (2NF).
If we see relations, we can say that, all non-key attributes are fully functional dependent on one attribute of the relation named primary key.

### 5.3.3 Third Normal Form

A relation that is in second normal form and has no transitive dependencies present is in third normal form. There should be no situation that a non-key attribute's value can be obtained by knowing the value of another non-key attribute. If we see our relations, we can say that, there is no situation in any table to get same result from non-key attribute as primary key can get.

## 5.4 Definition of data types and constraints

Now, after normalization process, we have completed the process of getting good designed and principled tables. Now we will represent the tables according to their data types and constraints. This representation of the tables is almost the same as exist in any practical database.

### 5.4.1 WORKFLOW

```
Name                              Null?    Type
--------------------------------- -------- --------
WF_ID                             NOT NULL VARCHAR2(20)
VERSION                           NOT NULL NUMBER
DISPLAY_NAME                               VARCHAR2(20)
```

Table 5.1 Workflow

### 5.4.2 ACTIVITY

```
 Name                             Null?    Type
 -------------------------------- -------- --------
 WF_ID                            NOT NULL VARCHAR2(20)
 ACTIVITY_ID                      NOT NULL VARCHAR2(20)
 VERSION                          NOT NULL NUMBER
 RESULT_TYPE_ID                            VARCHAR2(20)
 DISPLAY_NAME                     NOT NULL VARCHAR2(20)
 DESCRIPTION                               VARCHAR2(20)
 START_END                                 VARCHAR2(20)
 PERFORMER                                 VARCHAR2(20)
 DURATION                                  NUMBER
 ACTIVITY_TYPE                             VARCHAR2(20)
 MESSAGE_CONTENTS                          VARCHAR2(200)
 MESSAGE_DISPLAY_VALUE                     VARCHAR2(20)
 NODE_GEOMETRY                             VARCHAR2(20)
```

Table 5.2 ACTIVITY

### 5.4.3 ACTIVITY_ATTRIBUTES

```
Name                            Null?    Type
------------------------------- -------- -------
WF_ID                           NOT NULL VARCHAR2(20)
ACTIVITY_ID                     NOT NULL VARCHAR2(20)
VERSION                         NOT NULL NUMBER
ATTRIBUTE_ID                    NOT NULL VARCHAR2(20)
ATTRIBUTE_TYPE                  NOT NULL VARCHAR2(20)
```

Table 5.3 ACTIVITY_ATTRIBUTES

### 5.4.4 ACTIVITY_STATUSES

```
Name                            Null?    Type
------------------------------- -------- -------
INSTANCE_ID                     NOT NULL NUMBER
WF_KEY                          NOT NULL VARCHAR2(20)
VERSION                         NOT NULL NUMBER
WF_ID                           NOT NULL VARCHAR2(20)
ACTIVITY_STATUS                 NOT NULL CHAR(18)
ACTIVITY_ID                              VARCHAR2(20)
PERFORMER                                VARCHAR2(20)
```

Table 5.4 ACTIVITY_STATUSES

### 5.4.5 ACTIVITY_TRANSITIONS

```
Name                            Null?    Type
------------------------------- -------- -------
FROM_PROCESS_ACTIVITY           NOT NULL NUMBER
RESULT_CODE                     NOT NULL VARCHAR2(20)
TO_PROCESS_ACTIVITY             NOT NULL NUMBER
ARROW_GEOMETRY                           VARCHAR2(20)
```

Table 5.5 ACTIVITY_TRANSITIONS

### 5.4.6 ACT_DOC

```
Name                            Null?    Type
------------------------------- -------- -------
DOC_ID                          NOT NULL VARCHAR2(20)
WF_ID                           NOT NULL VARCHAR2(20)
ACTIVITY_ID                     NOT NULL VARCHAR2(20)
VERSION                         NOT NULL NUMBER
```

Table 5.6 ACT_DOC

### 5.4.7 ATTRIBUTES

```
Name                            Null?    Type
------------------------------- -------- -------
WF_ID                           NOT NULL VARCHAR2(20)
VERSION                         NOT NULL NUMBER
ATTRIBUTE_ID                    NOT NULL VARCHAR2(20)
DISPLAY_NAME                    NOT NULL VARCHAR2(20)
VALUE_TYPE                      NOT NULL VARCHAR2(20)
```

Table 5.7 ATTRIBUTES

### 5.4.8 ATTRIBUTE_VALUES

```
Name                            Null?    Type
------------------------------- -------- -------
ATTRIBUTE_ID                    NOT NULL VARCHAR2(20)
INSTANCE_ID                     NOT NULL NUMBER
WF_KEY                          NOT NULL VARCHAR2(20)
VERSION                         NOT NULL NUMBER
WF_ID                           NOT NULL VARCHAR2(20)
ATTRIBUTE_TYPE                  NOT NULL VARCHAR2(20)
ATTRIBUTE_TEXT_VALUE                     VARCHAR2(20)
ATTRIBUTE_NUMBER_VALUE                   NUMBER
ATTRIBUTE_DATE_VALUE                     DATE
```

Table 5.8 ATTRIBUTE_VALUES

### 5.4.9 EMPLOYEE

```
Name                            Null?    Type
------------------------------- -------- -------
EMP_ID                          NOT NULL VARCHAR2(20)
EMPNAME                         NOT NULL VARCHAR2(20)
PASSWORD                        NOT NULL VARCHAR2(20)
```

Table 5.9 EMPLOYEE

### 5.4.10 PROCESS_ACTIVITIES

```
Name                            Null?    Type
------------------------------- -------- -------
INSTANCE_ID                     NOT NULL NUMBER
WF_ID                                    VARCHAR2(20)
VERSION                                  NUMBER
ACTIVITY_ID                              VARCHAR2(20)
PERFORMER                                VARCHAR2(20)
```

Table 5.10 PROCESS_ACTIVITIES

### 5.4.11 RESULT_CODE

```
Name                            Null?    Type
------------------------------- -------- -------
WF_ID                           NOT NULL VARCHAR2(20)
VERSION                         NOT NULL NUMBER
RESULT_TYPE_ID                  NOT NULL VARCHAR2(20)
RESULT_CODE                     NOT NULL VARCHAR2(20)
```

Table 5.11 RESULT_CODE

### 5.4.12 RESULT_TYPE

```
Name                            Null?    Type
------------------------------- -------- -------
WF_ID                           NOT NULL VARCHAR2(20)
VERSION                         NOT NULL NUMBER
RESULT_TYPE_ID                  NOT NULL VARCHAR2(20)
```

Table 5.12 RESULT_TYPE

### 5.4.13 WF_USERS

```
Name                            Null?    Type
------------------------------- -------- -------
WF_KEY                          NOT NULL VARCHAR2(20)
WF_ID                           NOT NULL VARCHAR2(20)
VERSION                         NOT NULL NUMBER
PERFORMER                       NOT NULL VARCHAR2(20)
EMP_ID                          NOT NULL VARCHAR2(20)
```

Table 5.13 WF_USERS

### 5.4.14 WORKFLOW_ITEMS

```
Name                            Null?    Type
------------------------------- -------- -------
WF_KEY                          NOT NULL VARCHAR2(20)
WF_ID                           NOT NULL VARCHAR2(20)
VERSION                         NOT NULL NUMBER
OWNER                           NOT NULL VARCHAR2(20)
```

Table 5.14 WORKFLOW_ITEMS

# Chapter 6
# IMPLEMENTATION
# IN J2SE & J2EE

# 6. Implementation

The goal of development and implementation phase is to create the logic of the system design that will translate into the code in a specific language. For a given design the aim in this phase is to implement the design in the best possible manner. The development and implementation phase affects both testing and maintenance profoundly. A well written algorithm and logic can reduce the testing and maintenance effort. An important concept that helps the understandability of program is structured programming and object oriented programming. The goal of the structured programming is to make the control flow in the program linear, where as object oriented programming is based on events, functions and state of the object.

In this project Secret Service designed in object oriented so coding will perform in object oriented programming.

## 6.1 Technology Used

We have developed this new enterprise system in Java Technology. It would be very helpful before going into the software details if we provide very brief idea regarding this technology.

Java technology readily harnesses the power of the network because it is both a programming language and a selection of specialized platforms. As such, it standardizes the development and deployment of the kind of secure, portable, reliable, and scalable applications required by the networked economy.

The Java language is a programming language (much like C and C++). Built upon the Java language are a number of "editions": The **J2xE**'s. There are 3 editions of Java, first two closely related and the other less.

- Java 2 Standard Edition (J2SE)
- Java 2 Enterprise Edition (J2EE)
- Java 2 Micro Edition (J2ME)

J2EE, J2SE and J2ME are basically different specifications. These editions specify which parts of the Java language runtime features must be supported and which programmer API's are available.

### 6.1.1 Java 2 Standard Edition (J2SE)

Java 2 Standard Edition 1.4 (J2SE 1.4) is the specification of the Java language, version 1.4. For example the J2SE 1.4 contains a whole set of API's for socket manipulation and GUI building. Then we will see optional packages like Advanced Imaging, Cryptography etc.

### 6.1.2 Java 2 Enterprise Edition (J2EE)

J2EE is a superset of J2SE. That means it contains more classes than the J2SE. It is built on top of the J2SE and uses the same tools and compilers. J2SE is the "foundation" of the java language. If we write something using J2SE we can run it on a machine with J2EE but if we write something using J2EE we can't run it on a matching with only J2SE. The later will lack some classes that we use.

These are other packages that were put together for server-side, for companies. J2EE includes some additional tools for creating and developing things like Enterprise Java Beans, Remote Objects, Java Server Pages, Servlets, Java Naming and Directory Interface etc.

### 6.1.3 Java 2 Micro Edition (J2ME)

J2ME is a subset of J2SE so everything we write in J2ME should run under J2SE but not the other way around.

Java 2 Micro Edition (J2ME) is only used for cellular phones, PDAs programming.

### 6.1.4 Java Development Kit (JDK)

The Java Development Kit 1.4 is Sun's reference implementation of the Java 2 Standard Edition 1.4 specification.

## 6.2 Class Description

**Class name:** Activity

**Description:** Use to create the activity instances of a Workflow

Given below is detail of class exposed with their description.

```
package wfclient;

import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.sql.•;
import dms_viewer.*;

public class Activity {

  String id = "";        // Unique name of Activity
  String label = "";     // Display name of Activity
  String desc = "";      // Brief Description of Activity
  String performer = "";       // Like Clerk, Manager
  String message = "";  // The message that performer will forward
  String START_END = new String("MIDDLE");
  String geometry = ""; // This will contain the icon's geometry
  private boolean isCreated = false;
  LinkedList sentAttrib = new LinkedList();
  LinkedList recievedAttrib = new LinkedList();
  Vector comp_attatched = new Vector();
  int doc_comp_index = -1;
  String resultType;
  String act_type = new String();
  String MessageDname = new String("Message name");

  public Activity() {}

  public Activity(String id) {
    this.id = id;
  }

  public Activity(String id, String disply, String str) {
    this.id = id;
    this.label = disply;
    this.START_END = str;

  }
```

```
public Activity(String id, String label, String desc, String
    performer, String message, String START_END, int time, String
    geometry, LinkedList sentAttrib, LinkedList recievedAttrib,
    String resultType, String act_type, String MessageDname) {
  this.id = id;
  this.label = label;
  this.desc = desc;
  this.performer = performer;
  this.message = message;
  this.START_END = START_END;
  this.time_out = time;
  this.geometry = geometry;
  this.sentAttrib = sentAttrib;
  this.recievedAttrib = recievedAttrib;
  this.resultType = resultType;
  this.act_type = act_type;
  this.MessageDname = MessageDname;
  this.isCreated = true;
}

public void addDocComponent(Doc_Comp comp) {
  String docid = comp.getTempID();

  if (this.alreadyExists(docid)) {
    this.comp_attatched.remove(this.doc_comp_index);
    this.comp_attatched.add(this.doc_comp_index, comp);
  }
  else {
    this.comp_attatched.add(comp);
  }

}

public void removeDocComponent(Doc_Comp comp) {
  String docid = comp.getTempID();

  if (this.alreadyExists(docid)) {
    this.comp_attatched.remove(this.doc_comp_index);
  }

}

public Vector getDocComponent() {
  return this.comp_attatched;
}

public boolean alreadyExists(String doc_id) {
  boolean ret = false;
  for (int index = 0; index < this.comp_attatched.size(); index++) {
    Doc_Comp dctemp = (Doc_Comp)this.comp_attatched.get(index);

    if (doc_id.equals(dctemp.getTempID())) {
      doc_comp_index = index;
      ret = true;
      return ret;
    }
  }
```

```
    return ret;
  }

  public Vector getComponentVector() {
    return this.comp_attatched;
  }

  public boolean saveActivity(Connection conn, String wfid, int ver) {
    boolean ret;
    try {

      PreparedStatement activity_stmt;

      String activity = "insert into
           ACTIVITY(WF_ID,VERSION,ACTIVITY_ID," +
          "RESULT_TYPE_ID,DISPLAY_NAME,DESCRIPTION,PERFORMER," +
          "START_END,DURATION,ACTIVITY_TYPE,MESSAGE_CONTENTS,MESSAGE_
           DISPLAY_VALUE,NODE_GEOMETRY)" +
          " VALUES ( ?,?,?,?,?,?,?,?,?,?,?,?,?)";

      activity_stmt = conn.prepareStatement(activity);

      activity_stmt.setString(1, wfid);
      activity_stmt.setInt(2, ver);
      activity_stmt.setString(3, this.id);
      activity_stmt.setString(4, this.resultType);
      activity_stmt.setString(5, this.label);
      activity_stmt.setString(6, this.desc);
      activity_stmt.setString(7, this.performer);
      activity_stmt.setString(8, this.START_END);
      activity_stmt.setInt(9, this.time_out);
      activity_stmt.setString(10, this.act_type);
      activity_stmt.setString(11, this.message);
      activity_stmt.setString(12, this.MessageDname);
      activity_stmt.setString(13, this.geometry);


      activity_stmt.executeUpdate();
      // Save Doc Components attached
      String act_doc =
          "INSERT INTO ACT_DOC(DOC_ID, WF_ID, ACTIVITY_ID, VERSION)" +
          "VALUES(?,?,?,?) ";
      String doc_comp = "INSERT INTO DOC_COMP(COMP_ID,
      DOC_ID,WF_ID,ACTIVITY_ID,VERSION) VALUES (?,?,?,?,?)";

      PreparedStatement act_doc_stmt = conn.prepareStatement(act_doc);
      PreparedStatement doc_comp_stmt =
      conn.prepareStatement(doc_comp);

      for (int i = 0; i < this.comp_attatched.size(); i++) {
        Doc_Comp tempDC = (Doc_Comp)this.comp_attatched.get(i);
        String doc_id = tempDC.getTempID();
        Vector tempComp = tempDC.getComponents();

        act_doc_stmt.setString(1, doc_id);
        act_doc_stmt.setString(2, wfid);
        act_doc_stmt.setString(3, this.id);
        act_doc_stmt.setInt(4, ver);

        act_doc_stmt.executeUpdate();

        for (int j = 0; j < tempComp.size(); j++) {
          String comp = (String) tempComp.get(j);

          doc_comp_stmt.setString(1, comp);
          doc_comp_stmt.setString(2, doc_id);
          doc_comp_stmt.setString(3, wfid);
          doc_comp_stmt.setString(4, this.id);
          doc_comp_stmt.setInt(5, ver);


          doc_comp_stmt.executeUpdate();

        }
      }
      ret = true;
    }
    catch (Exception exp) {
      exp.printStackTrace();
      ret = false;
      Object[] options = {
          "OK"};
      JOptionPane.showOptionDialog(null, "Record not inserted in
      ACTIVITY","Error", JOptionPane.DEFAULT_OPTION,
      JOptionPane.ERROR_MESSAGE, null, options, options[0]);

      //Now close the connection
      try {
        conn.rollback();
        conn.close();
      }
      catch (Exception ex) {
        System.out.println(ex.toString());
      }

    }
    return ret;
  }

  public void setSentAttrib(String id) {
    sentAttrib.addLast(id);
  }

  public void setSentAttrib(Collection c) {
    sentAttrib = new LinkedList(c);
  }

  public String getSentAtrrib(int i) {
    return (String) sentAttrib.get(i);
  }

  public void setSentAttrib(String[] id) {
    for (int i = 0; i < id.length; i++)
      sentAttrib.addLast(id[i]);
  }

  public LinkedList getSentAttrib() {
    return sentAttrib;
  }

  public void setRecievedAttrib(String id) {
    recievedAttrib.add(id);
  }

  public void setRecievedAttrib(Collection v) {
    recievedAttrib = new LinkedList(v);
  }

  public String getRecievedAttrib(int index) {
    return (String) recievedAttrib.get(index);
  }

  public void setRecievedAttrib(String[] id) {
    for (int i = 0; i < id.length; i++)
      recievedAttrib.add(id[i]);
  }

  public LinkedList getRecievedAttribList() {
    return recievedAttrib;
  }

  public void setResultType(String rt) {
    this.resultType = rt;
  }

  public String getResultType() {
    return this.resultType;
  }

  public void setID(String id) {
    this.id = id;
  }

  public String getID() {
    return id;
  }

  public void setLabel(String id) {
    label = id;
  }

  public String getLabel() {
    return label;
  }

  public void setDesc(String id) {
    this.desc = id;
  }

  public String getDesc(String id) {
    return desc;
  }

  public void setMessage(String str) {
    this.message = str;
  }

  public String getMessage() {
    return this.message;
  }

  public void setPerformer(String str) {
    performer = str;
  }

  public String getPerformer() {
    return this.performer;
  }

  public void setTimeout(int time) {
    time_out = time;
  }

  public int getTimeout() {
    return time_out;
  }

  public void enableCreated(boolean b) {
    isCreated = b;
  }

  public boolean isCreated() {
    return isCreated;
  }

  public void setNodeGeometry(String b) {
    geometry = b;
  }

  public void setNodeGeometry(Point p1) {
    geometry = p1.x + "," + p1.y;
  }

  public LinkedList getSendAttributes() {
    return this.sentAttrib;
  }

  public LinkedList getReceavedAttributes() {
    return this.recievedAttrib;
  }

  public String getNodeGeometry() {
    return geometry;
  }

}
```

# Chapter 7

# TESTING

# 7. Testing

Testing is the process of executing a program with the explicit intention of finding errors i.e., making the program fail and test cases are devised with the purpose in mind. A test case is a set of data items that the system processes as normal input. A successful test is the one that finds an error. Software testing is partly intuitive but largely systematic. Good testing involves much more than just running the program a few times to see whether it works.

There are some important types of testing as given below:

- Glass Box Testing
  - Focused testing
  - Control Flow
  - Data Integrity
  - Module / Unit or Element Testing
  - Integration Testing
  - Internal Boundaries Testing
  - Algorithm-Specific Testing
- Black Box Testing
  - Functional Testing
  - Big Bang Testing
  - Performance Testing
  - Regression Testing
  - Beta Testing
  - Port Testing
  - Quality Testing

- Load Testing
- Stress Testing

## 7.1 Glass Box Testing

The testing in which a tester has access to the code is called glass box or structural testing. The programmer usually does it. It has further testing types.

### 7.1.1 Focused Testing

The programmer can test the program in pieces. He can write special test code that feeds interesting values to an isolated module, and reports intermediate results obtained from the module. I tested my application modules by giving interesting values. e.g. We tested **"Add Activity"** module by giving it different values. It shows proper messages on any error or operation.

### 7.1.2 Control Testing

The programmer knows that what the system is supposed to do next. He can modify the program so that it constantly reports what its id doing, or he can use a special program called debugger to run the program and track the order in which lines of code are executed.

Using Borland JBuilder 8 Enterprise Edition, we debugged whole application to control flow test.

### 7.1.3 Data Integrity

The programmer knows which parts of the program modify its item of data. He might also write special code that calculates the value that a test variable should have at a given point in the program.

### 7.1.4 Module/ Unit/ Element Testing

In this testing different modules are checked independent of their functionality. Even the part is performing its task in the program correctly, but not coronet in sense is reported as a bug, because it can rise at any other time.

We tested my DPSFLOW as standalone application. When I run it, a stand alone application runs.

We choose the New Workflow from File Menu. Default application pane opens with start and end activity placed already. It means that user must start his workflow from start activity and end it to end activity.

Similarly we tested all components of DPSFLOW.

### 7.1.5 Integration Testing

The modules are not integrated with a sequence but different modules are integrated and then checked their functionality. we did integration testing by integrating Activity Module and Relation module.

### 7.1.6 Internal Boundaries Testing

The programmer can see internal boundaries in the code that is completely invisible to the outside tester. Let a program use so temporary storage so the programmer is in the good position to test that at which points it should do what, i.e. what maximum storage it can do.

### 7.1.7 Algorithm - Specific Testing

In this testing the good or bad behaviors of the program are not examined but algorithm is checked. Let a program have to use memory so efficiently so whenever it is algorithmically tested the main focus will be that how does it manages the memory. Similarly, if a program has to do some animations on the screen then its performance for animation will be checked, it will not be checked that how it is managing the memory.

## 7.2 Black Box Testing

In this type of testing the code is invisible from the tester; he can just check it from outside of it. Black box testing has following forms.

### 7.2.1 Functional Testing

Different parts of the software are tested and checked the behavior. Let there are two modules one makes the file and other module reads it at another time. So at first when the first module has completed its execution then immediately the file is deleted and checked that what happens with the next module. In this way the internal functionality of program is checked from outside of it.

### 7.2.2 Big Bang-Testing

In this testing different parts of the software are run without any sequence, and checked that is their any failure in the system. This testing has some bad features.

- It is too hard to figure out what caused the failure.
- Error in one module can also block other module.
- Bad feeling during testing.

The simple way of it is just do mouse click non-sequentially on the application system screen. We tested application by pressing different function and other keys while application starts. We checked that it might hit some place cause a failure, but a. Secondly, run different options of the program at a time, it might cause error in the system.

### 7.2.3 Performance Testing

It works like Algorithm specific testing; the clear difference is that in that testing we can see the code, while in it we can not see it. We guess outside the system that what is happening inside.

### 7.2.4 Regression Testing

In it the test cases are repeated again and again. It itself is not a testing but a mechanism. Let open miscellanea address dialog, enter some wrong values, it will give error messages. Now close it and open it again, enter values. Repeat testing again and again. This mechanism will ensure that there are no bugs in address dialog.

### 7.2.5 Beta Testing

It is testing neither done by programmer nor by the quality-assuror, but the software is released as a beta version of the system. Now the users of the system use it find its errors and report it to the company. It has two benefits. Firstly, the software is used by the people that might unaware of the system that's why they will be able to find more simple errors. Secondly, it tells that in which direction the work should be done.

At present we have not released this software so beta testing was not possible.

### 7.2.6 Port Testing

In this testing phase the system is checked for portability, it contains

- Over all functionality
- Key board handling
- Disks
- Operating system error handling
- Installation
- Compatibility
- Interface
- Other changes that programmer knows well.

### 7.2.7 Load Testing

In this testing the program is tried to run at its maximum. We checked our application by creating heavy Workflow that contained many Activities it runs smoothly. There is some change in interface but rest of the application runs proficiently.

### 7.2.8 Stress Testing

In this testing the program is tried to run while many other applications are also running and the processor is continuously busy. Let run a media file + a web server + a database server + word + excel + power point + antivirus + … and continuously check that what our system is doing as the memory and processor is going to be busy more and more. Our System worked fine.

# Chapter 8
# USER MANUAL

# 8 User manual

This manual provides the detail explanation of DPSFlow (Web-Enabled Workflow Management System). DPSFlow has three modules.

- Workflow Builder
- Web Admin
- Workflow Clients/Users Module

First one is Stand alone application that is called Workflow Builder, which provides graphical user interface to the user to create and/or modify workflow processes. These processes are stored in database.

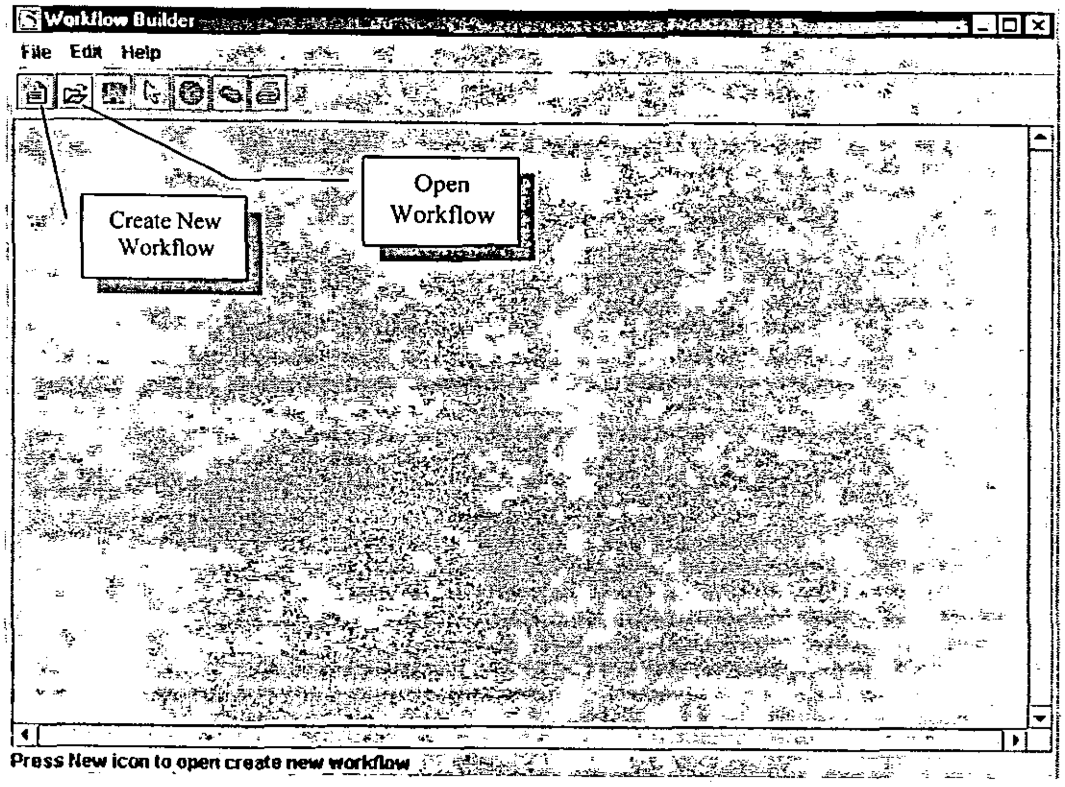
Second module is web based application that uses application server or web server to run and manipulate the stored workflow processes. Following are detailed interfaces for this complete application.

## 8.1 Workflow Builder

This is stand alone part of the Web-Enabled Workflow Management System which can run on any platform. It uses Oracle Thin Drivers to connect to database in which schema resides.

## 8.1.1 Create New Workflow

To create new workflow in workflow builder, he will first click on new button on the tool bar in figure 8.1. Then he will see a default model which is shown in figure 8.2 .



Press New icon to open create new workflow

Figure 8.1 Workflow Builder just after launching application

Now the disabled icons have become active. User will create activities and relate them accordingly. We will discuss each step in detail later. Here is the default layout of new workflow model. This contains two main features. One is create new activity and second is relate activities.

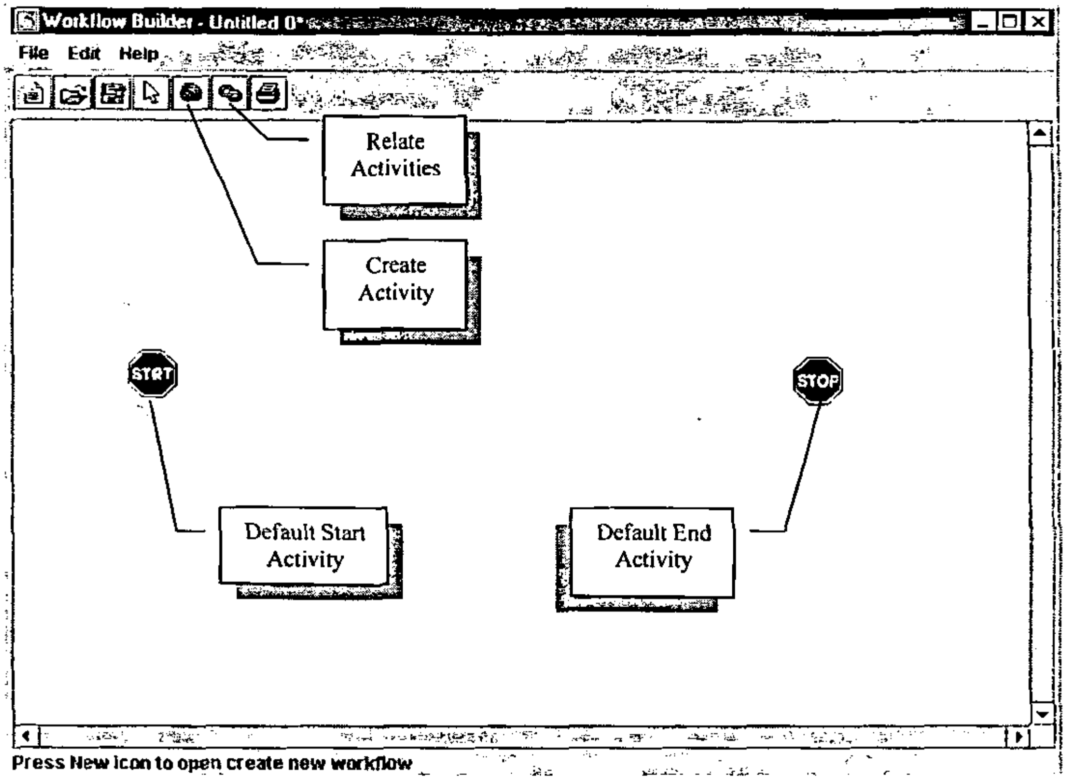


Figure 8.2 Workflow Builder default layout

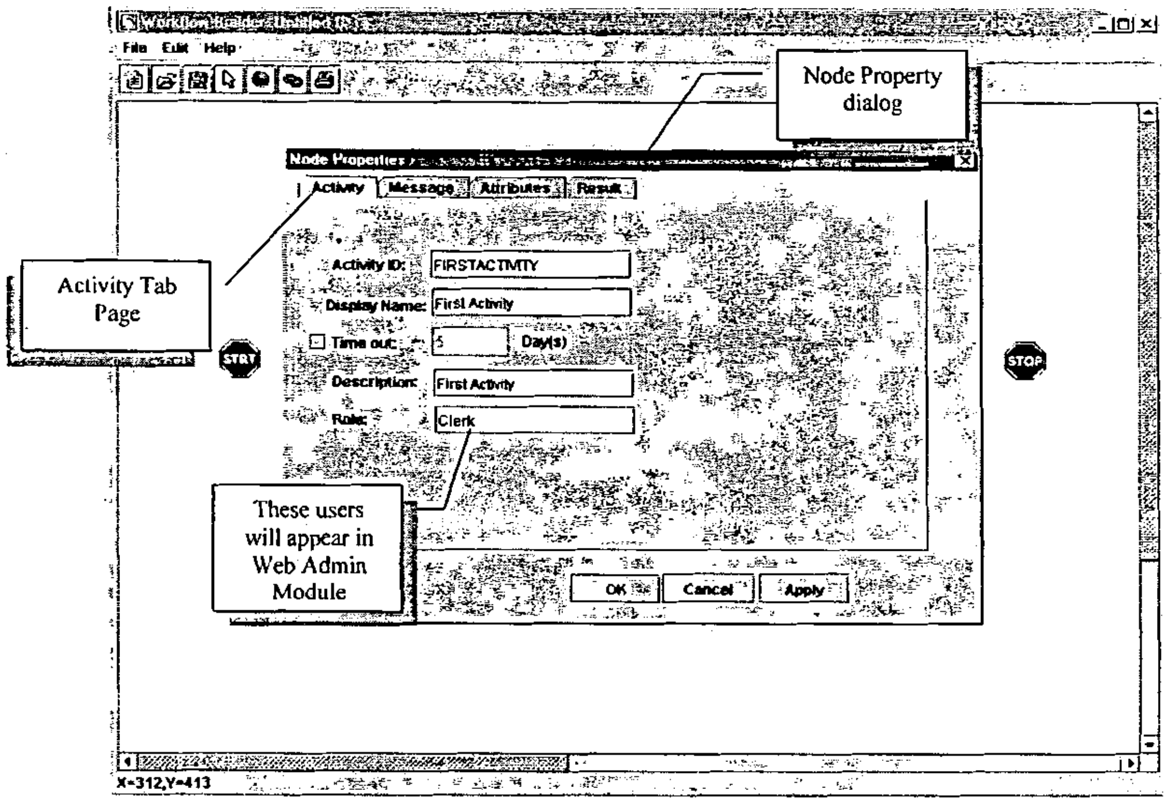
Whenever a new workflow process is made it should start from default start node and must end on default end node.

Now to create new activity user will click on create activity icon on the tool bar and then will click on main panel. A new dialog will open that is shown in figure 8.3. This property dialog further contains further tab pages which are explained in detail below.

### 8.1.2 Node Properties - Activity Tab Page

This form takes necessary inputs from the user who is creating new activity. User must have to fill Activity ID input screen without blank spaces and Display Name. This Display name option for future work that this will help programmers to internationalize Workflow Builder.

In this screen workflow user is specified like clerk, manager etc



Activity ID: FIRSTACTIVITY
Display Name: First Activity
Time out: 5 Day(s)
Description: First Activity
Role: Clerk

Figure 8.3 Node Properties - Activity *(continued)*

### 8.1.3 Node Properties - Attributes Tab Page

Any activity can send any number of attributes to next activities. The flow will be managed by the connection of activities. When the user will create attributes those attributes will be stored in the Global Attribute Container. After creating all the attributes user will select an attribute then he will click send able or receivable button to assure that whether the user of this activity will send the value of this attribute to the next user or will it receive the values in these attributes provided that some other user send him these attributes. This will·be more clear when we will explain the web interface.

Following is the main screen for defining and selecting new attributes

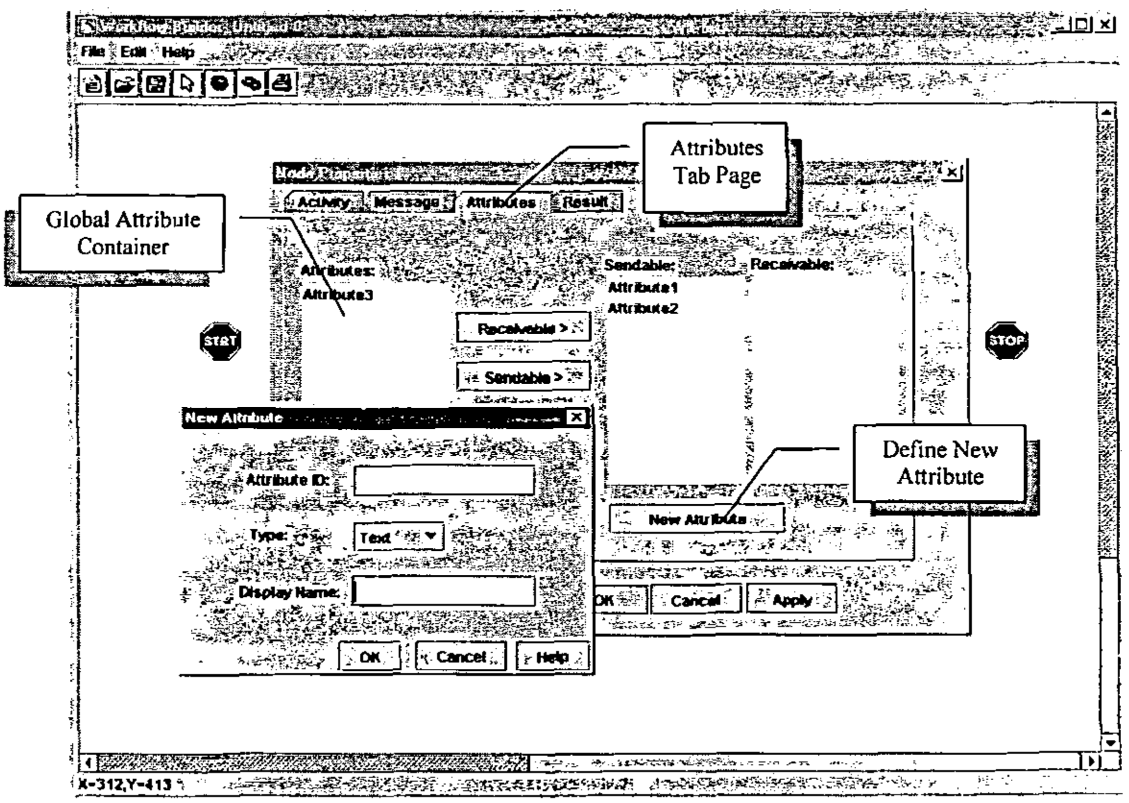


Figure 8.4 Node Properties - Attribute *(continued)*

### 8.1.4 Node Properties - Result Tab Page

Finally user will specify that what are the possible paths that can be followed by this activity by specifying the result type and result codes. For example *"Clerk"* is assigned an activity like *"Verify Account"*. Now there can be two possible paths that can be followed on the result of Clerk's activity. He can approve user's contents or can reject them. It means that he should see two buttons on the screen, **Approved & Reject.** So here, in Workflow Builder, while defining the workflow, we will create one group id that is called *Result Type,* like **APPROVE_REJECT** which is result type. Then we will create unique paths, called *Result Codes*, like one will be **Approve,** second will be **Reject.**

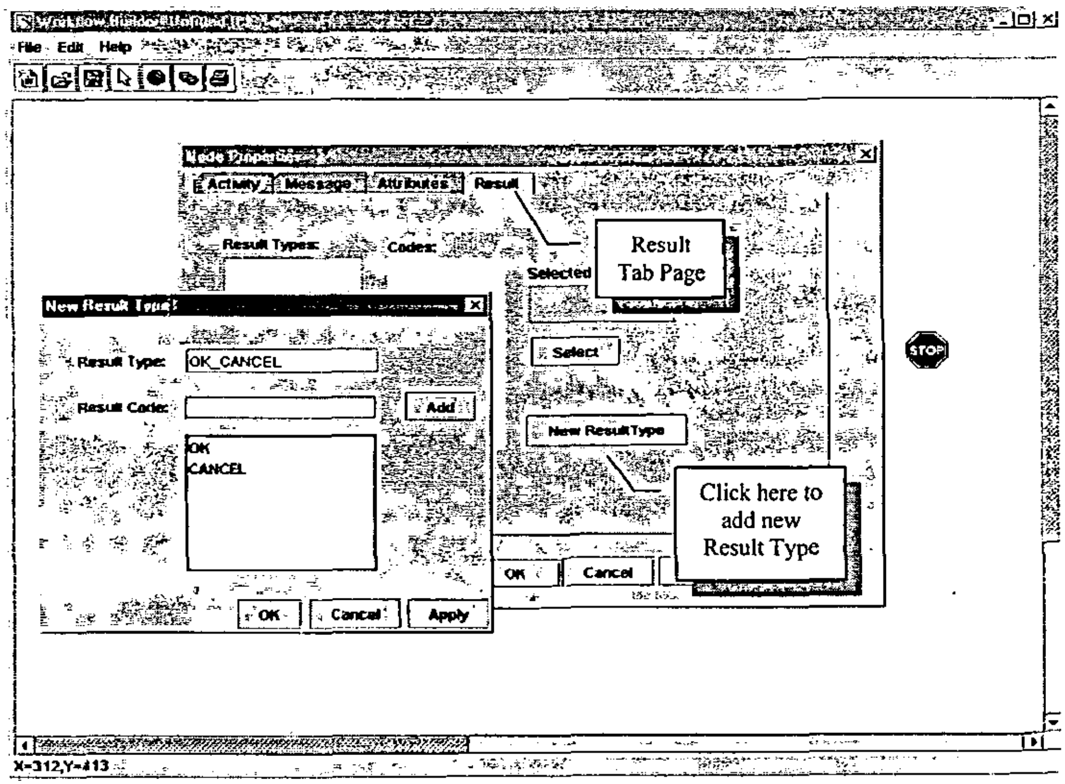


Figure 8.5 Node Properties – Result *(continued)*

Finally when the user will press OK button then the activity will be placed on the client panel, which is shown in the following diagram with the label "First Activity" Similarly we create all activities and then we join all activities with connecting tool.

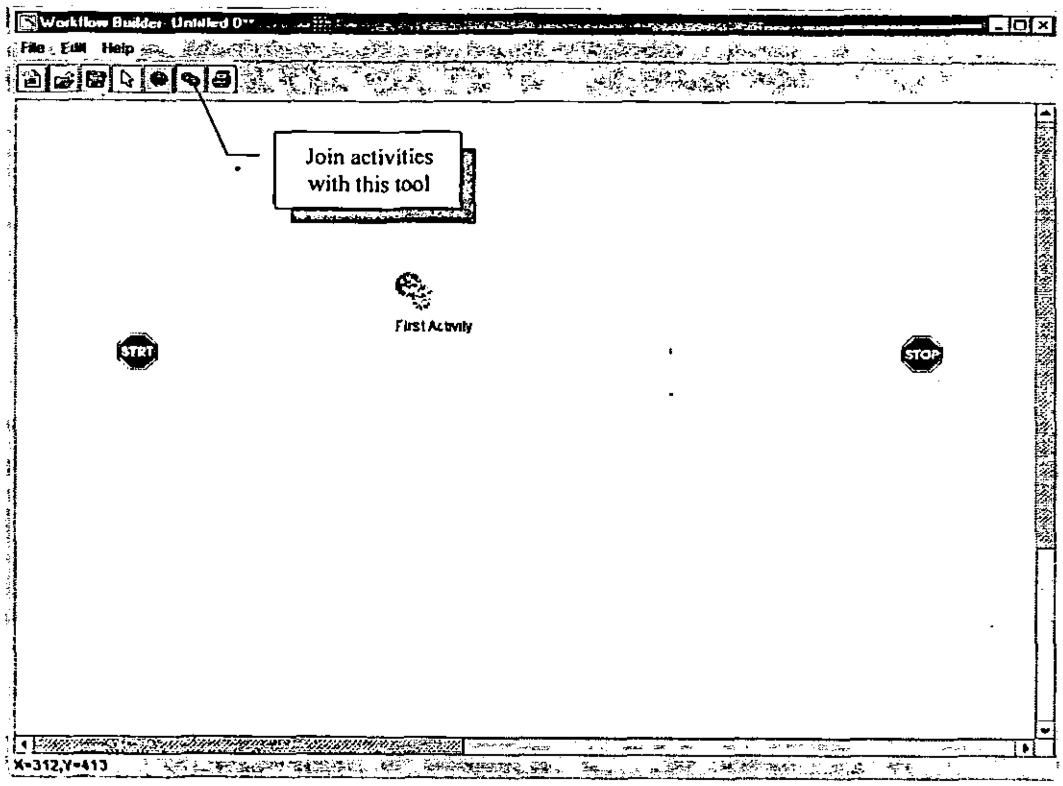
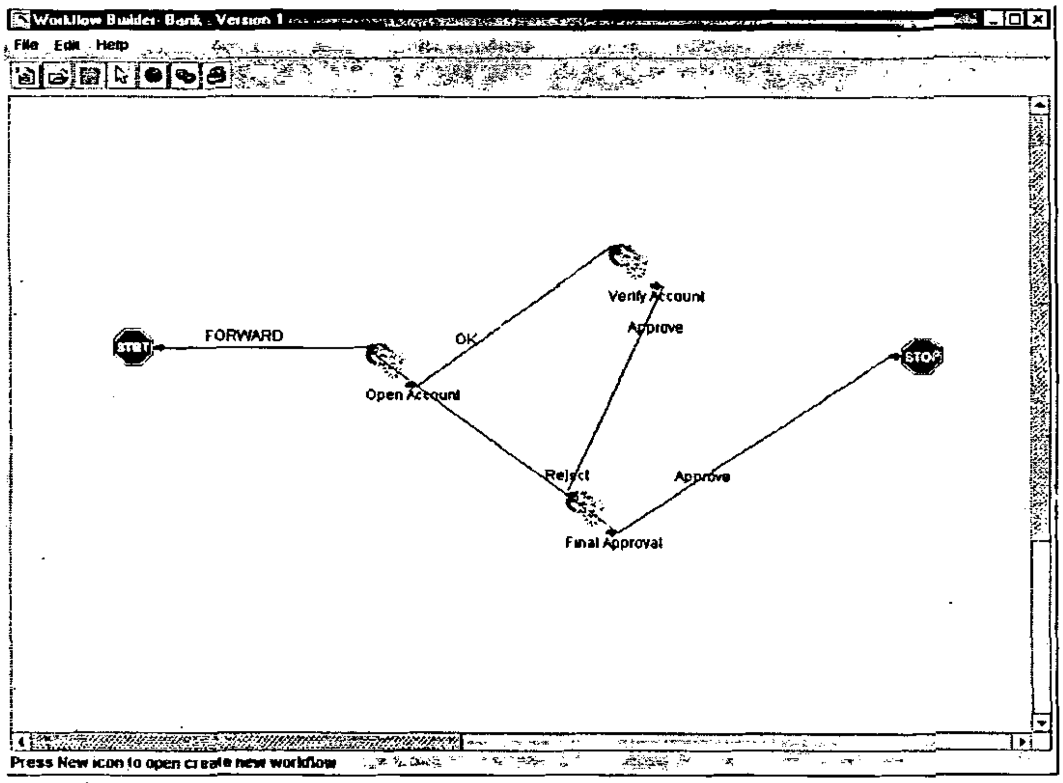


Figure 8.6 Node Properties *(continued)*

We have created a sample Workflow named "Bank" which is shown in the figure 8.7 below.



Press New icon to open create new workflow

Figure 8.7 Complete Workflow

Then this Workflow will be saved in database and we will move to web based interface of this system.

## 8.2 Web Admin Module

This is our web based interface for the Workflow Management System. Administrator will use this part. This interface is used to Launch and maintain the workflow.

### 8.2.1 Login

Administrator will provide his id and password to enter in the system. The screen shot is shown below

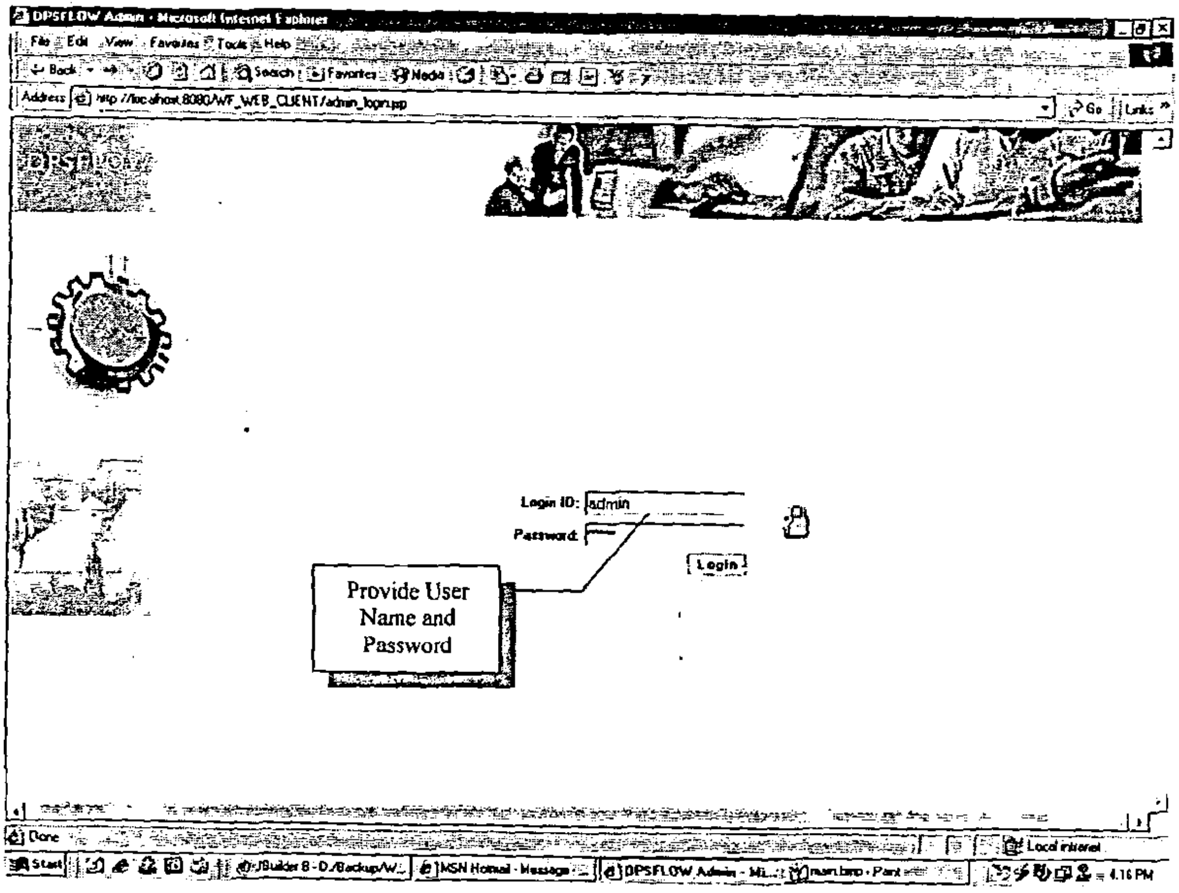

DPSFLOW Admin - Microsoft Internet Explorer
http://localhost:8080/WF_WEB_CLIENT/admin_login.jsp
Login ID: admin
Password:
Login
Provide User Name and Password

Figure 8.9 login screen for administrator

### 8.2.2 Select Workflow

Now administrator can view the workflows that were created using Workflow Builder. He will click on the link *Launch New Workflow*. All workflows along with their versions will be shown. Now he will click on show details button to view the details of the particular workflow, which is shown in figure 8.11.

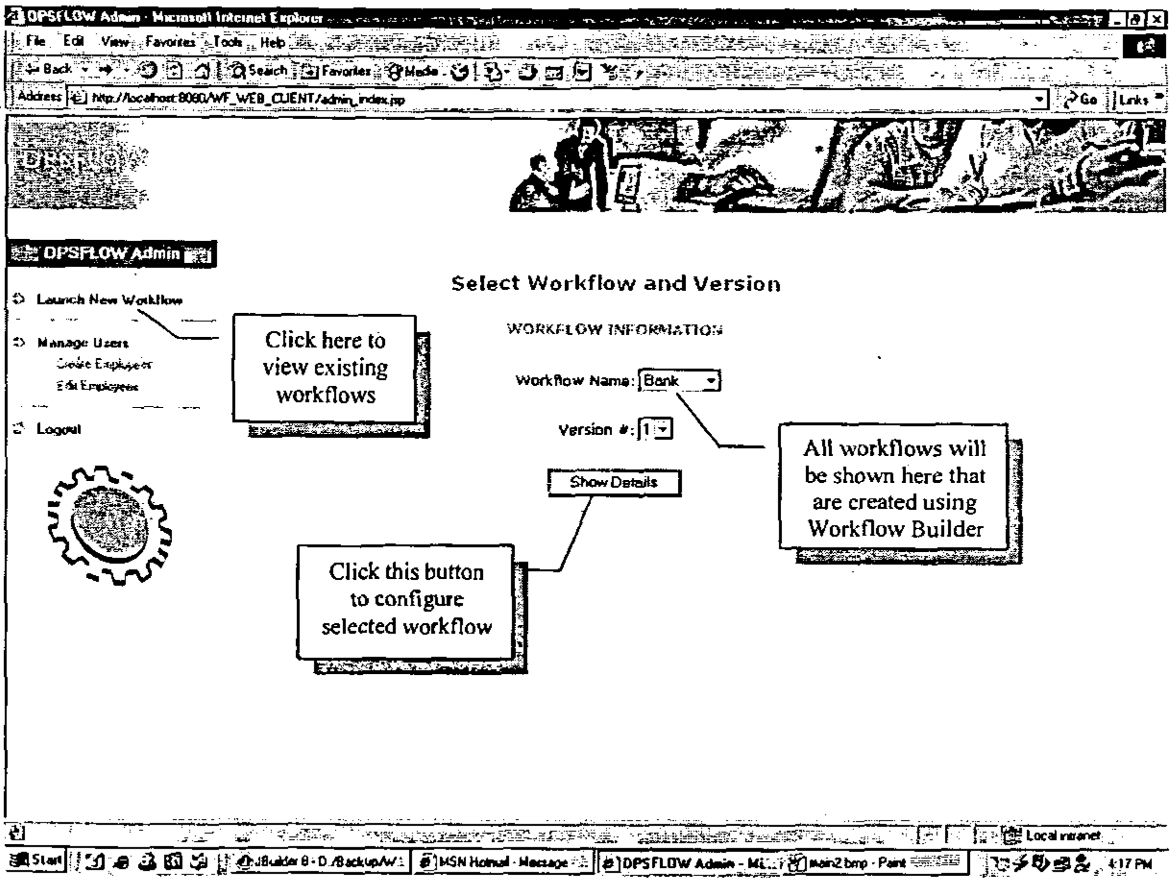
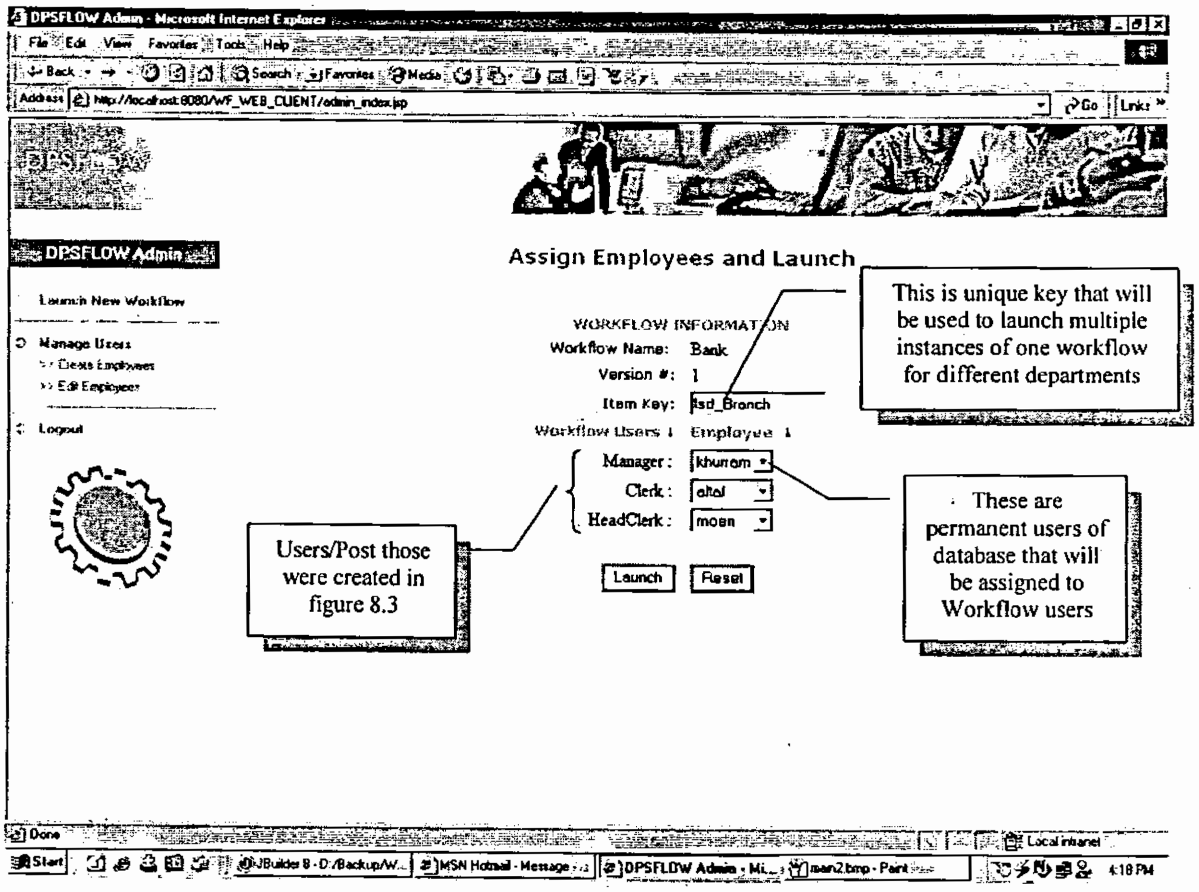

DPSFLOW Admin - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://localhost:8080/WF_WEB_CLIENT/admin_index.jsp
DPSFLOW Admin
Launch New Workflow
Manage Users
Create Employee
Edit Employee
Logout
Select Workflow and Version
WORKFLOW INFORMATION
Workflow Name: Bank
Version #: 1
Show Details
Click here to view existing workflows
All workflows will be shown here that are created using Workflow Builder
Click this button to configure selected workflow
Local intranet

Figure 8.10 Launch Workflow on the Web *(continued)*

### 8.2.3 Launch Workflow

Now administrator will see the details of the workflow selected in figure 8.10. In this interface he will see all users that are actually workflow users like Clerk, Manager, Peon etc. These users can be created on the fly while defining new workflow. Then they will be assigned permanent database roles whose privileges can also be managed. These Roles can also be created and maintained.

This is also possible that we have created a workflow named *"Purchase Request Flow"* that is used to purchase and PC in Accounts Department. Now if Examination Department says that they also want to use this Workflow then we will have to Launch the same Workflow for them. It means that there should be some mechanism to uniquely identify the same workflow for two different departments. Here we have introduced the word *"Item Key"* Just give the unique key for Examination Branch which must be different from the key used by Accounts Department.



DPSFLOW Admin - Microsoft Internet Explorer

Address: http://localhost:8080/WF_WEB_CLIENT/admin_index.jsp

Assign Employees and Launch

WORKFLOW INFORMATION

Workflow Name: Bank

Version #: 1

Item Key: Isd_Branch

Workflow Users : Employee :

Manager : khurram

Clerk : altaf

HeadClerk : moen

This is unique key that will be used to launch multiple instances of one workflow for different departments

Users/Post those were created in figure 8.3

These are permanent users of database that will be assigned to Workflow users

Figure 8.11 Launch Workflow on the Web – Details of Selected Workflow

## 8.3 Workflow Clients/Users Module

This is third interface of DPSFlow which is used by the clients or the participants of the workflow. We will explain this in detail.

### 8.3.1 Client Login

Same like administrator, the workflow participant will be logged in the system by providing the correct user name and password. The screen shot is shown below.

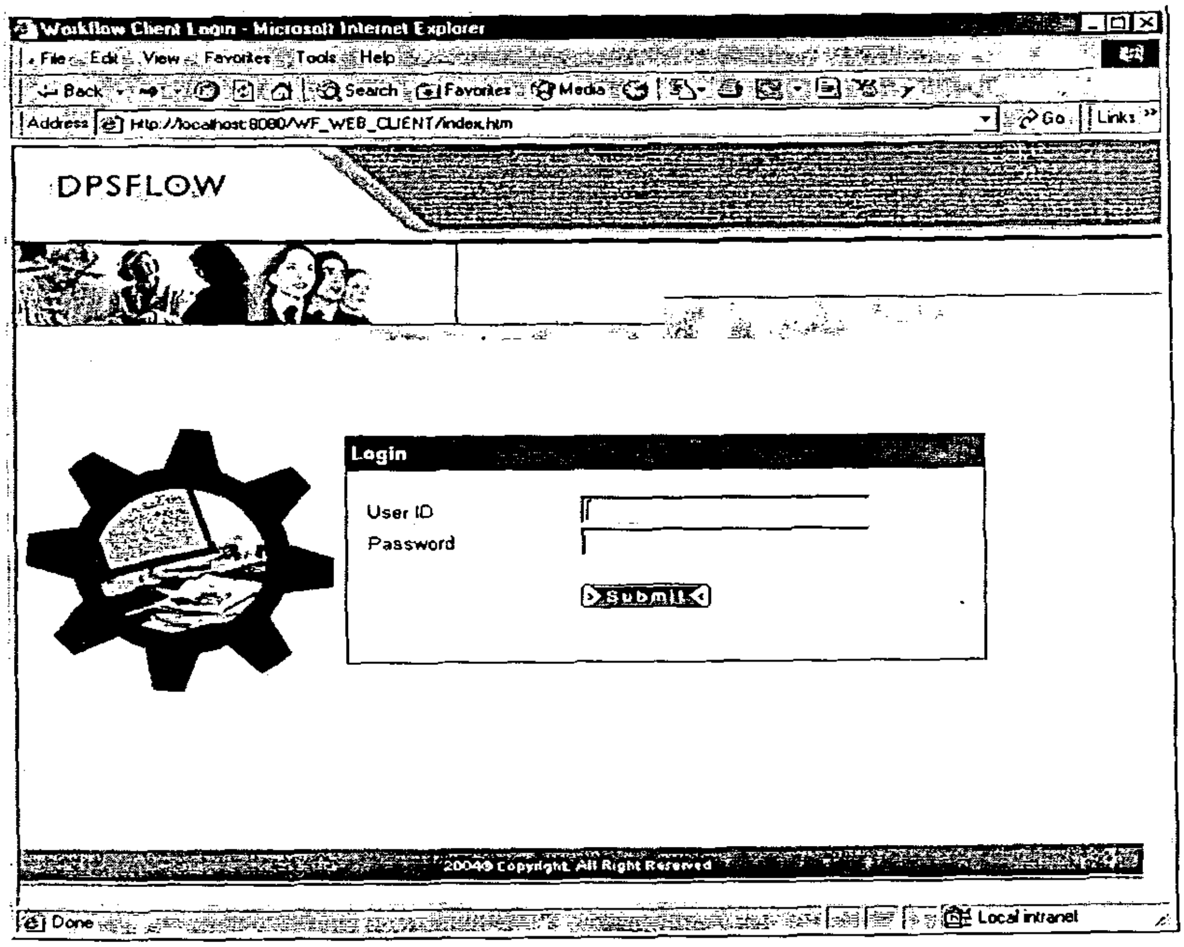
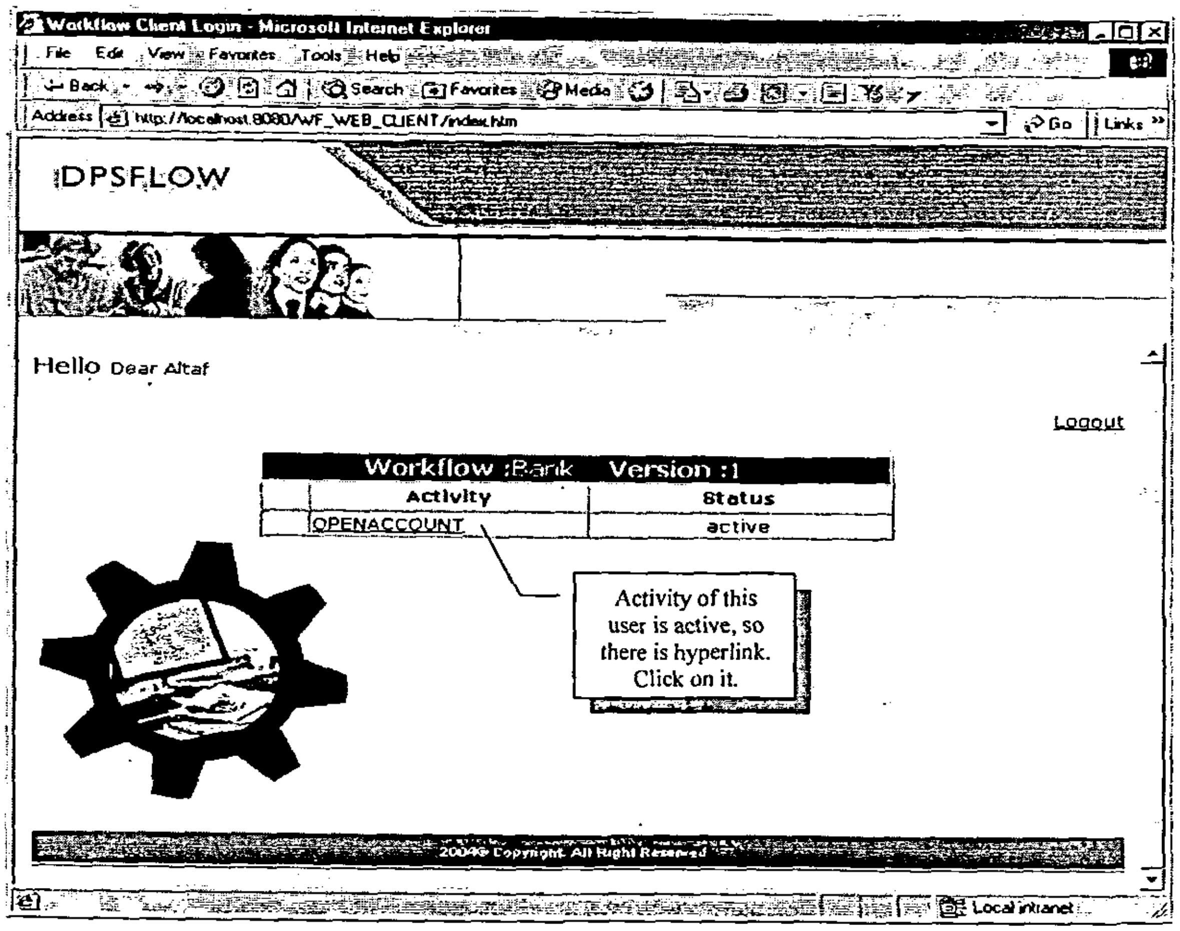

Workflow Client Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address Http://localhost:8080/WF_WEB_CLIENT/index.htm
Go Links
DPSFLOW
Login
User ID
Password
Submit
2004© Copyright All Right Reserved
Done
Local intranet

Figure 8.12 login screen for Workflow User/Participant

## 8.3.2 Activity Statuses

After successful login the user will see the status of its activity. Now below is the first user of our sample workflow named "*Bank*" If the second user is also logged in the system its status will be shown ready and its activity will be disabled. It means that until the first user will not perform its activity the second user will not be allowed to perform its activity task.

Now user will click its activity *"OPENACCOUNT"*. A new screen will be opened for the user showing the details of the activity. That is shown in figure 8.14


Workflow Client Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/WF_WEB_CLIENT/index.htm
Go Links
DPSFLOW

Hello Dear Altaf

Logout

| Workflow :Bank | Version :1 |
|---|---|
| Activity | Status |
| OPENACCOUNT | active |

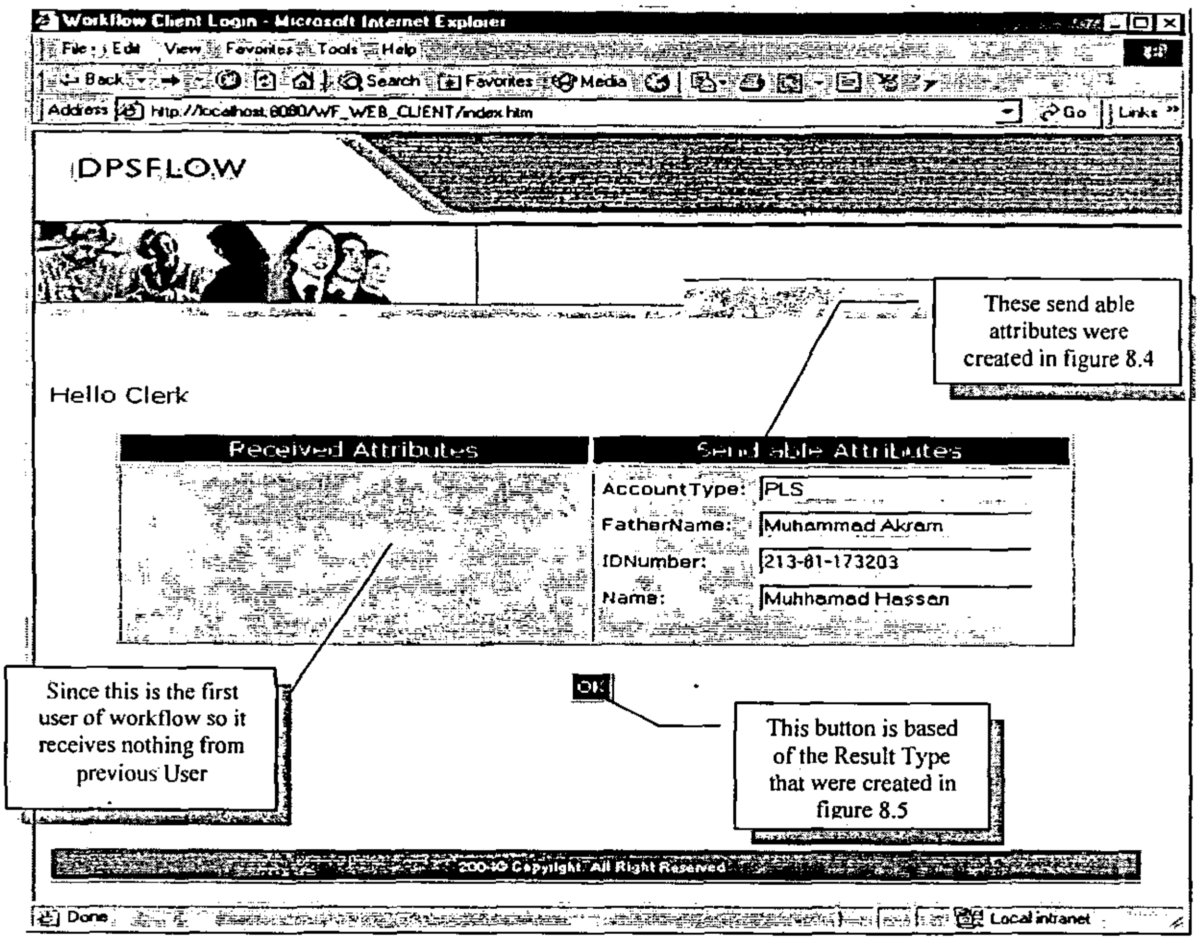
Activity of this user is active, so there is hyperlink. Click on it.

2004© Copyright All Right Reserved
Local intranet

Figure 8.13 Activity Status for Workflow User/Participant

### 8.3.3 Detail of Activity for User One

After clicking the hyperlink in figure 8.13, the user will see following screen. Here he will see two panes showing Received Attributes and Send able Attributes. He will fill the values in Send able attributes and will press appropriate button to complete its activity. (After viewing figure 8.15, this process will be easy to understand)


Workflow Client Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/WF_WEB_CLIENT/index.htm
Go Links
DPSFLOW
These send able attributes were created in figure 8.4
Received Attributes
Send able Attributes
Since this is the first user of workflow so it receives nothing from previous User
OK
This button is based of the Result Type that were created in figure 8.5
2004© Copyright. All Right Reserved
Done
Local intranet

Hello Clerk

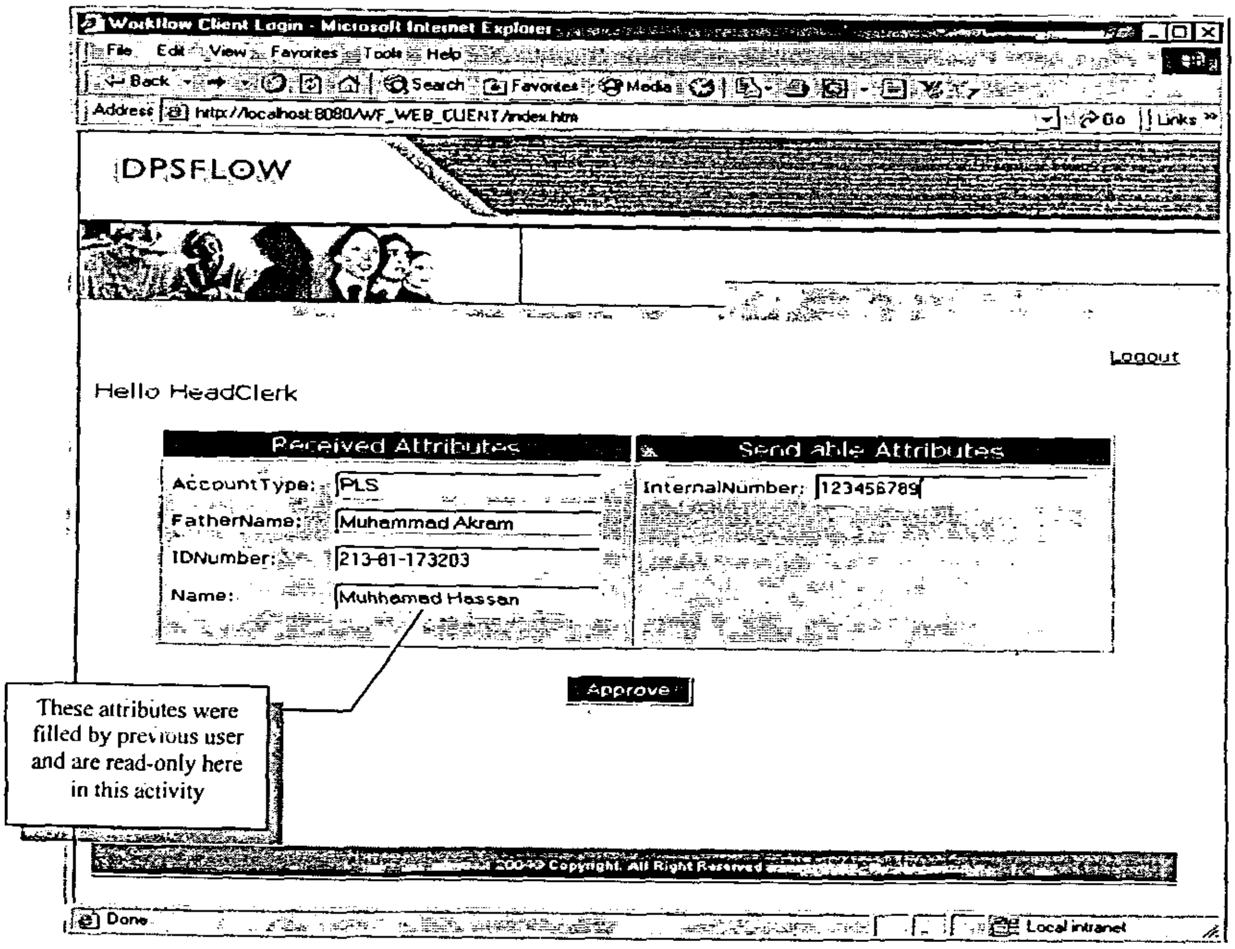
| | |
|---|---|
| AccountType: | PLS |
| FatherName: | Muhammad Akram |
| IDNumber: | 213-81-173203 |
| Name: | Muhhamad Hassan |

Figure 8.14 Detail of Activity for Workflow User/Participant

### 8.3.4 Detail of Activity for User Two

Now when the second user will view its Status Active, he will enter its activity. He will see following screen. After pressing Approve button he will forward control to next user and his activity will be ended.

Similarly this process will continue until last user completes its activity and then the workflow will be ended and we will again have to launch it.


Workflow Client Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/WF_WEB_CLIENT/index.htm
Go Links
DPSFLOW
Logout
Approve
2004© Copyright. All Right Reserved
Done
Local intranet

Hello HeadClerk

| Received Attributes | | Send able Attributes | |
|---|---|---|---|
| AccountType: | PLS | InternalNumber: | 123456789 |
| FatherName: | Muhammad Akram | | |
| IDNumber: | 213-81-173203 | | |
| Name: | Muhhamad Hassan | | |

These attributes were filled by previous user and are read-only here in this activity

Figure 8.15 Detail of Activity for Workflow User/Participant

# BIBLIOGRAPHY AND REFERENCES

# Bibliography and References

[1] Workflow Management Coalition Interface 1: Process Definition Interchange Process Model; Document Number WfMC TC-1016-P
Document Status - Version 1.1 (Official release) Issued on October 29, 1999

[2] Workflow Management Coalition Interface 2: Workflow Client Application Application Programming Interface (WAPI) Specification; Document Number WFMC-TC-1009 October-97 Version 2.0e (Beta)

[3] S. Jablonski, C. Bussler: Workflow Management Modeling Concepts, Architecture and Implementation, London, International Thomson Computer Press, 1996.

[4] Document Management and Workflow: A Case Study Marius P Jooste Vega Consulting Solutions Inc.

[5] Khalid A. Mughal, Rolf W. Rasmussen : A Programmers Guide to Java Certification Addison Wesley, 2001

[6] Gustavo Alonso, C. Mohan; Workflow Management Systems: The Next Generation Of Distributed Processing Tools

## RESEARCH PAPER

NETWORK FOR SCIENTIFIC INFORMATION

16-May-2005

Dr. Khurram Shahzad
International Islamic University
Islamabad
pakistan

Subject: **Galley proof of Article No. 68-ITJ**

Dear Dr. Khurram Shahzad,

We hope you will receive this letter with good health.

Please find enclosed herewith galley proof of above mentioned manuscript for final checking. Please point out the corrections clearly on the manuscript and also provide the typed corrections on a separate sheet also. Your quick response will help us to accommodate your article in the coming issue.

According to our record printing cost is still laying pending. It is therefore, requested to please send printing cost with the checked galley proof.

ANSInet cordially invites to please visit the Journal's website http://www.ansinet.net

Waiting for your quick response.

Regards

[illegible]

Muhammad Imran
Executive Managing Editor

308-Lasani Town, Sargodha Road, Faisalabad, Pakistan
Tel: 0092-0412001145-46 Fax: 0092-21-5206036 E-mail: support@ansinet.org http://www.ansinet.org

Information Technology Journal 4 (): CCCC, 2005
ISSN 1812-5638
© 2005 Asian Network for Scientific Information

# A Web-enabled Architecture of Workflow Management System for Heterogeneous Environment

[1]Khurram Shahzad and [2]Khalid Rashid
[1]Department of Computer Sciences, [2]Faculty of Applied Sciences,
International Islamic University, Islamabad, Pakistan

**Abstract:** In this study architectural framework of workflow management system for heterogeneous and distributed environment has been proposed which incorporates web-enabled independent interface for clients to execute workflows. Some of the drawbacks and limitations of the traditional approaches are discussed; then architecture of flexible and platform independent simple workflow management system has been presented which is based on Java and internet technologies. Workflow engine and clients are implemented in Java. Workflow models are stored in relational database and the workflow engine accesses these models using JDBC interface. Standard browsers are used as web based clients to access the workflow system via HTTP protocol.

**Key words:** Business process, workflow, activity, process execution, workflow engine, transition, workflow participant

## INTRODUCTION

Workflow management is one of the areas that in recent years have attracted the attention of many researchers, developers and users. It has primarily evolved from the need to understand, to organize and, often ultimately, to automate the processes on which a business is based. Workflow is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve an overall business goal. This computerized facilitation or automation of a business process, in whole or part is called workflow. Where as a system that completely defines, manages and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic is called Workflow Management System[1].

A workflow is an abstraction of business process. It consists of activities, which correspond to individual process steps and agents, which execute these activities. An agent may be an information system (e.g., a database system), a human (e.g., a customer representative) or combination of both (e.g., a human using a software program). The workflow specification may also specify execution durations for both the activities and the process itself[2].

## WORKFLOW MANAGEMENT

The basic concepts of workflow management system can be best introduced using the definitions provided by the reference model of the Workflow Management Coalition (WFMC), an international organization leading the efforts to standardize workflow management product. It defines workflow as:

The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules[1].

Workflow management includes a build time and a run time dimension imposing different requirements each[3]. The build time requirements comprise all issues related to modeling of control and data flow, like graphical or formal representation, ease of use, readability, abstraction and modularity, or correctness (Fig. 1).

The run time requirements are related to the activation, enactment and termination of workflows. The system should be extendible to be able to handle future unforeseen application scenarios. Workflow definitions may change over time and thus workflows should be dynamically customizable and adaptable. Continuously growing application areas and increasing organizational coverage call for a scalable system. The system should be open that it can run in a distributed heterogeneous computing infrastructure[3].

Workflow management aims at modeling and controlling the execution of process in both business and scientific applications. It has gained increasing attention recently, since it allows combining data-oriented view on applications, which is traditional for an information system, with a process-oriented one in which activities and their occurrence over time are modeled and supported properly[2].

**Corresponding Author:** Khurram Shahzad, Department of Computer Sciences, International Islamic University, Islamabad, Pakistan Tel: 051 4540668 or 051 90 166671 E-mail: ksiiui@yahoo.com

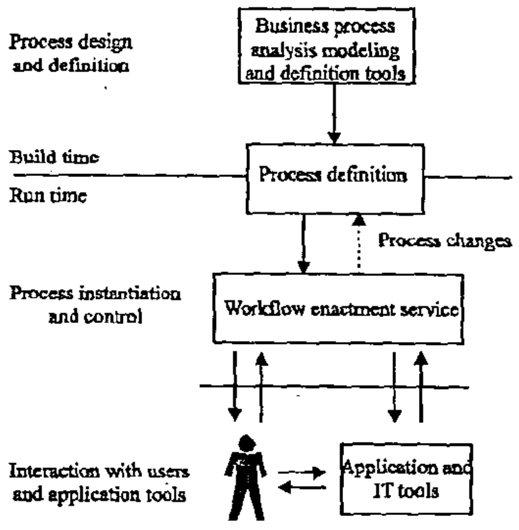

Process design and definition
Business process analysis modeling and definition tools
Build time
Run time
Process definition
Process changes
Process instantiation and control
Workflow enactment service
Interaction with users and application tools
Application and IT tools

Fig. 1: Workflow system characteristics

## CURRENT STATE OF WORKFLOW MANAGEMENT SYSTEMS

According to the Workflow Management Coalition (WFMC) reference model[1], a workflow consists of an engine, application agents, and administration and monitoring tools (Fig. 2). The process definition tool is visual editor which is used to define the specification, i.e., the schema, of workflow process. The same schema can be used for creating multiple instances of the same business process at a later time. The workflow engine and the tools communicate with a workflow database to store and update workflow relevant data, such as workflow schemas, statistical information, and control information required to execute and monitor the active process instances[2].

## LIMITATIONS OF EXISTING SYSTEMS

The state of the art in workflow management has been determined so far by the functionality provided in commercial systems. Paradoxically, this has been the major source of limitations. Many products were developed without clear understanding of the user requirements and these products were quiet unprepared to meet the demands placed upon them by eager users. Issues such as performance, scalability or reliability are hardly ever considered in these areas, an unfortunate characteristic inherited by the workflow products.

Current workflow systems are not designed to be embedded into earlier applications easily. This causes problem for large scale enterprise deployments, independent software vendors and system integrators who need comprehensive business process management facility for their applications. Current homegrown, quick and dirty workflow engines are dead-end.

Moreover existing systems are almost incompatible. In spite of the efforts of the workflow management coalition, current products incorporate in the design very

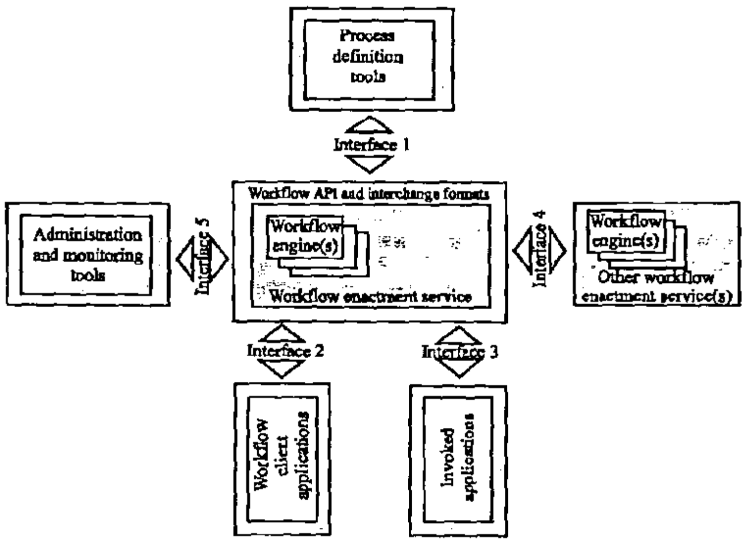

Process definition tools
Interface 1
Workflow API and interchange formats
Workflow engine(s)
Workflow enactment service
Administration and monitoring tools
Interface 5
Interface 4
Workflow engine(s)
Other workflow enactment service(s)
Interface 2
Interface 3
Workflow client applications
Invoked applications

Fig. 2: WFMC reference architecture

concrete and exclusive interpretations of the world that makes particularly impossible to federate different systems. These incompatibilities are not just the syntax or the platform, but the very interpretation of the workflow execution. In most cases, the workflow is so tied to the underlying support system that it is not feasible to extend its functionality to accommodate other workflow interpretations[4].

Workflow specifications may be ambiguous or contain inflexible semantics. Visual workflow programming languages sometimes lack representations for timing and execution constraints of workflows[5].

## SYSTEM ARCHITECTURE

This workflow management system provides support in two functional areas. Build time and Runtime control. The *build time functions* support the definition and modeling of workflow processes. Internally all elements of workflow engine/server can reside on the same node or be transparently distributed across many nodes. This architecture can grow incrementally from a small compact system to a large system that can handle hundreds of processes per hour.

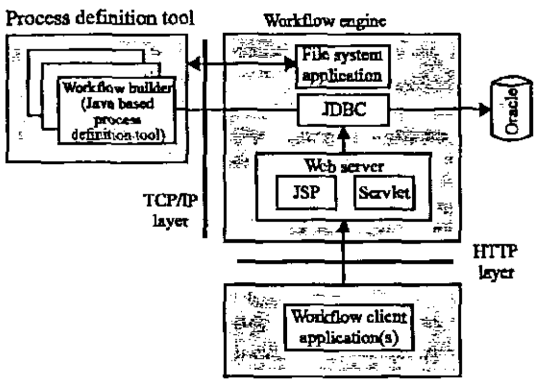


Fig. 3: System architecture

The Runtime control functions are implemented in workflow client interface, which handle the execution of processes. Runtime control is built on top of persistence storage which allows system to recover from failure. In this recovery process data is not lost like if it is present in live thread. Navigational logic of end user between activities governs workflow. Within the workflow management system two components are considered, storage server and navigation server. These are referred to as the workflow engine. Navigation server is also called web server. This interacts with users as well as with other invoked applications (Fig. 3).

Workflow client functionality is delivered through a web browser. Web server reads workflow models from database, controls the execution of workflows and then generates web based interfaces related to the activity for specific end user and renders it to him. Users access the workflow system using web based client.

## SYSTEM IMPLEMENTATION

Following design decisions have been adopted to achieve main goals of flexibility and platform independence.

1. Thin client architecture has two parts; Process definition tool, named Workflow Builder and the HTTP based clients. Workflow Builder is written in Java. Java byte code can be interpreted on a large variety of platforms including the Sun Solaris and Windows platforms. This allows to execute the code on a large variety of systems without recompiling it.
2. All other client functionalities like starting the process, modifying process at run time, access tasks and check status and history, is delivered through web browser.
3. Workflow models are currently stored in a relational database and the workflow server accesses these models using a JDBC interface[6]. Using this middleware component, different underlying relational database products can be used without changing the code.

## CONCLUSION AND FUTURE WORK

In this study current workflow management systems have been analyzed which lead us to conclude that current systems are inflexible, lack any standardization across any products and are not designed to be embedded into earlier application easily. Then architecture of flexible and platform independent simple workflow management system is presented for heterogeneous and distributed environment.

This proposed system incorporates all of the components necessary for adding process layer to business applications. The architecture is designed to be distributed on heterogeneous environment. Java is used as the workflow process definition tool and internet technologies are used as workflow client applications. This research and development is step towards building scalable and reliable distributed workflow management systems.

As already mentioned, workflow management can be considered an important direction in the future development of information systems. In future focus will be on reflexivity that allows a workflow component to programmatically examine, analyze, create, and manipulate its own process and data as part of automatable tasks within the executing workflow. We will evolve and optimize workflow management system through measurement, tracking, and reuse of generalizeable process fragments to improve the ability of the workflow to adapt new applications and uses.

## REFERENCES

1. David, H.W., 1995. The workflow reference model. Document No. TC00-1003. Document Status-Issue 1.1, Workflow Management Coalition 2 Crown Walk Winchester Hampshire, UK SO22 5XE, http://www.wfmc.org/standards/model.htm pp: 6-8.
2. Euthimous, P. and M. Rabinovich, 1997. Predictive workflow management. AT and T Labs-Research 180 Park Avenue Florham Park, NJ 07932, pp: 1-2.
3. Michael, W. and T. Illmann, 1998. Using java for the coordination of workflows in the world wide web. Faculty of Computer Science, University of Ulm, http://medien.informatik.uni-ulm.de/forschung/publikationen/interaktion98.pdf pp: 2-4.
4. Alonso, G., D. Agrawl, A. El. Abbadi and C. Mohin, 1996. Functionality and limitations of current workflow management systems. IEEE Expert J., pp: 14-17.
5. Peter, J.K., G.A. Bolcer and M. Bergman, 1998. Requirements for supporting dynamic and adaptive workflow on the WWW. University of California, Irvine. Irvine, CA http://citeseer.ist.psu.edu/kammer98requirements.html pp: 2-6.
6. Gottfried, V., M. Weske and G. Wittkowski, 1996. Dynamic workflow management on the web. http://citeseer.ist.psu.edu/vossen96dynamic.html pp: 10-15.