

2008

A Workflow Visual Modeler and Its Interface to Existing Workflow Management Systems

Jyoti Chaturvedi
University of North Florida

Suggested Citation

Chaturvedi, Jyoti, "A Workflow Visual Modeler and Its Interface to Existing Workflow Management Systems" (2008). *UNF Graduate Theses and Dissertations*. 187.
<https://digitalcommons.unf.edu/etd/187>

This Master's Thesis is brought to you for free and open access by the Student Scholarship at UNF Digital Commons. It has been accepted for inclusion in UNF Graduate Theses and Dissertations by an authorized administrator of UNF Digital Commons. For more information, please contact [Digital Projects](#).

© 2008 All Rights Reserved

A WORKFLOW VISUAL MODELER AND ITS INTERFACE TO
EXISTING WORKFLOW MANAGEMENT SYSTEMS

by

Jyoti Chaturvedi

A thesis submitted to the
School of Computing
in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

May, 2008

The thesis "A Workflow Visual Modeler and Its Interface to Existing Workflow Management Systems" submitted by Jyoti Chaturvedi in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the thesis committee:

Date

Signature deleted

12/11/07

Arturo Sanchez-Ruiz, Ph.D.

Thesis Advisor and Committee Chairperson

Signature deleted

12/10/2007

Robert F. Roggio, Ph.D.

Signature deleted

12/11/2007

Sherif A. Elmamy, Ph.D.

Accepted for the School of Computing:

Signature deleted

1/31/08

Judith L. Solano, Ph.D.

Director of the School

Accepted for the College of Computing, Engineering, and Construction:

Signature deleted

2/5/08

Neal S. Coulter, Ph.D.

Dean of the College

Accepted for the University:

Signature deleted

19 FEB 2008

David E. W. Fenner, Ph.D.

Dean of the Graduate School

ACKNOWLEDGEMENTS

First, I would like to thank Dr. Arturo Sánchez-Ruiz for accepting my request to be my thesis advisor and guide for this work. His continuous encouragement during the research and the time he dedicated towards it are worth more than just thanks.

In addition, I would like to thank my spouse for his support, during my year of work on this thesis. My children deserve a big thank you, as well, for their understanding of the importance of my work and for excusing my absence from some of their activities during this time.

CONTENTS

List of Figures	vi
List of Tables	viii
List of Abbreviations	ix
Abstract	x
Chapter 1: Introduction	1
Chapter 2: A Survey of Existing WfMSs	6
2.1 JBoss jBPM	6
2.2 Zebra	7
2.3 YAWL	8
2.4 XFlow	10
2.5 Enhydra JaWE	10
2.6 WFGGenSystem-UNF	11
Chapter 3: A Survey of Modeling Notations	14
Chapter 4: Diagramming Tools and Frameworks	19
4.1 JGraphPad Pro	19
4.2 ILOG JViews Diagrammer	20
4.3 Business Process Visual Architect	20
4.4 Enterprise Architect	21
4.5 Enhydra JaWE	21
4.6 Java Swing and Java 2D Frameworks	22
4.7 JGraph and JGraphLayout Frameworks	22

Chapter 5: The VM: Software Architecture and Implementation	24
Chapter 6: Case Studies	31
6.1 Case Study 1: WFGenSystem-UNF	32
6.2 Case Study 2: XFlow	39
Chapter 7: Conclusions and Future Work	44
7.1 Conclusions	44
7.2 Future Work	45
References	46
Appendix A: WfMSs, Diagramming Tools and Frameworks .	50
Appendix B: BPMN and UML Notation Comparison	51
Appendix C: JGraph XML Schema	61
Appendix D: WFGenSystem-UNF-Visual System Demonstration: UNF HelpDesk Example	63
Appendix E: XFlow System Demonstration: SimpleWorkflow Example	105
Vita	114

LIST OF FIGURES

Figure 1: WfMC Workflow Reference Model Taken from [WfMC95]	3
Figure 2: JBoss jBPM Graphical Process Designer Eclipse Plug-in	7
Figure 3: Zebra GUI Designer	8
Figure 4: YAWL Editor	9
Figure 5: Enhydra JaWE in Together Professional	11
Figure 6: Interaction Style Currently Used by WGenSystem-UNF to Design, Update and Modify a Workflow	12
Figure 7: VM Architecture	25
Figure 8: VM Toolbar	26
Figure 9: A Simple Workflow Using the Rich User Interface	27
Figure 10: Process Sub-Menu	28
Figure 11: Flow Sub-Menu	28
Figure 12: Workflow Sub-Menu	28
Figure 13: VM Architecture with Respect to Integration with WGenSystem-UNF and XFlow Applications	31
Figure 14: WGenSystem-UNF Architecture Diagram with the Integration of VM	33
Figure 15: Details of the VM Integration with WGenSystem-UNF	33
Figure 16: VM Integrated with WGenSystem-UNF	34

Figure 17: Workflow Properties Dialog	37
Figure 18: Process Properties Dialog	37
Figure 19: Flow Properties Dialog	38
Figure 20: VM Applet Integrated with XFlow	40
Figure 21: Workflow Properties Dialog for XFlow	42
Figure 22: XFLOW File Output from VM	43

LIST OF TABLES

Table 1: User Interface Packages	26
Table 2: Available Methods of Interface Class	29
Table 3: Classes of Package edu.unf.soc.vm.connectivity	35
Table 4: Classes of Package edu.unf.soc.vm.connectivity.db	35
Table 5: Classes of Package edu.unf.soc.vm.connectivity.dialog	36
Table 6: Classes of Package edu.unf.soc.vm.connectivity	41
Table 7: Classes of Package edu.unf.soc.vm.connectivity.db	41
Table 8: Classes of Package edu.unf.soc.vm.connectivity.dialog	41
Table 9: Methods Implemented for XFlow	42

LIST OF ABBREVIATIONS

AD	Activity Diagrams
API	Application Programming Interface
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMI	Business Process Management Initiative
BPMN	Business Process Modeling Notation
BP-VA	Business Process Visual Architect
GUI	Graphical User Interface
JaWE	Java Workflow Editor
OMG	Object Management Group
OASIS	Organization for the Advancement of Structured Information Standards
RFP	Request for Proposal
UML	Unified Modeling Language
VM	Visual Modeler
WfMC	Workflow Management Coalition
WfMS	Workflow Management System
XSD	XML Schema Definition
YAWL	Yet Another Workflow Language

ABSTRACT

The rapid growth and complexity of today's businesses have created a need for business process management approaches that will promote the efficient functioning of these organizations. Users of business process management tools greatly benefit from using visual process modeling capabilities. Cross-business interaction sets forth the need for standardization of notations in designing these models.

The goal of this thesis is to study state of the art business process management notations and state of the art diagramming frameworks associated with building a Visual Modeler that can be easily integrated with existing workflow management systems. This thesis presents a Visual Modeler that has been created based on the research findings. Two case studies are presented, which show how the modeler has been effectively integrated as part of two completely different workflow management systems.

Chapter 1

INTRODUCTION

A workflow can be simply defined as the routing of documents and/or tasks through a network of processes. The Workflow Management Coalition (WfMC) defines the term 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" [WfMC99].

Two categories of workflows are worth mentioning: scientific workflows and business workflows. The former is mostly concerned with routing of data through various algorithms, applications and services. The latter concentrates on business processes, including (among others) scheduling, and dependencies. These are not necessarily data-driven but rather are human-driven (e.g., to conduct a meeting).

A workflow management system (WfMS) is a software system that completely automates the definition, management, and execution of workflows. Typical elements of a WfMS include

a modeling component that enables administrators and analysts to define process and activities, an execution interface seen by end-users, and a workflow engine that performs the coordination of processes and activities.

The WfMC published the Workflow Reference Model [WfMC95] to characterize the major components and interfaces of a generic WfMS architecture (see Figure 1). The model suggests five interfaces a WfMS should support. Interface 1 defines process definition and modeling tools. Interface 2 defines progression of processes, activities, and work items in the client application. Interface 3 defines an invoked application interface that allows the workflow engine to invoke other applications. Interface 4 defines workflow enactment services to enable interaction with other workflow systems. Interface 5 defines administration and monitoring of the entire workflow system.

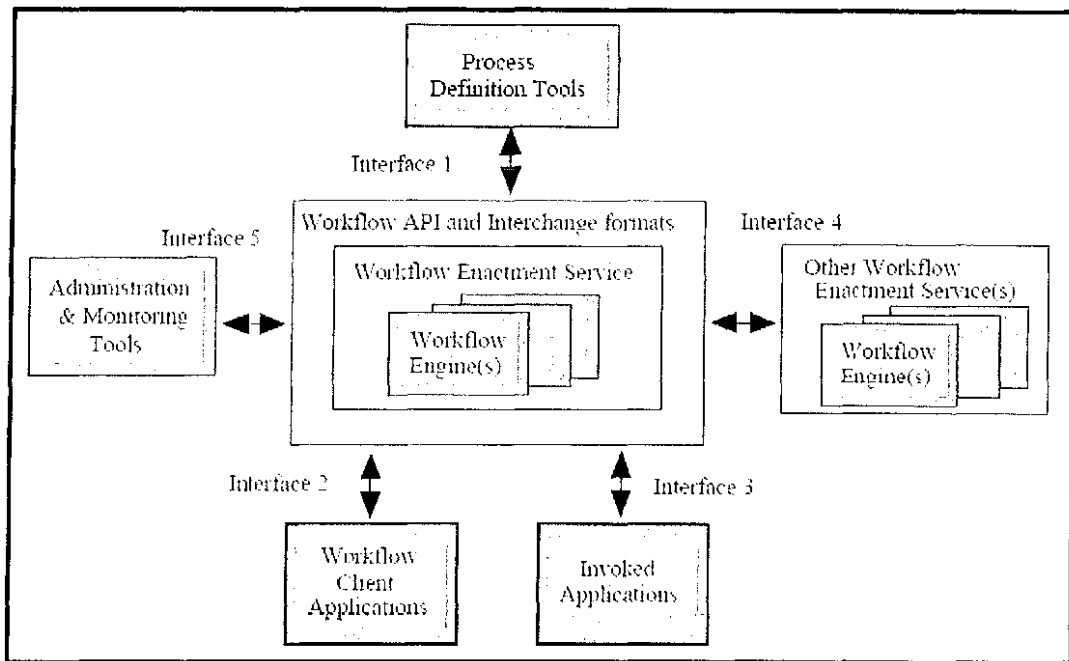


Figure 1: WfMC Workflow Reference Model Taken from [WfMC95]

The Workflow Generator and Tracking System (WFGenSystem-UNF)¹ is a generic WfMS, which allows end-users to define documents, workflow networks, and workflow systems. This system, which implements three of the five recommended interfaces, was developed by Stacy Hutchings, in cooperation with Dr. Arturo Sánchez, as part of her MS project [Hutchings05]. The three interfaces of the Workflow Reference Model, implemented in the WFGenSystem-UNF are Interface 1, Interface 2, and Interface 5. Currently, workflow engineers design/modify/update a workflow in WFGenSystem-UNF, by defining the states,

¹ See <http://orquidea.ccec.unf.edu:8080/WFGenSystem-UNF/>, if a loading exception is thrown ("Exception UNAVAILABLE"), please just press the "reload" button of your browser.

transitions, and outgoing/incoming strategies for the underlying process network, using a menu-driven interface.

This thesis deals with the construction of a visual workflow modeler and its integration with existing workflow systems through a well-defined architecture. A case study is presented which shows how the visual modeler was integrated with two completely different WfMS: WGenSystem-UNF and XFlow.

More specifically, preparing this thesis entailed:

1. Researching state of the art WfMS architectural blueprints.
2. Researching state of the art business process management (BPM) notations.
3. Researching state of the art diagramming frameworks.
4. Implementing a Visual Modeler (VM) using the research findings.
5. Interfacing the modeler with existing WfMS.
6. Deploying and testing the integrated software systems.

The rest of this document is organized as follows:
Chapter 2 covers a survey of existing WfMSs and their visual modeling capabilities. Chapter 3 reviews the business process modeling notations. Chapter 4 describes available diagramming frameworks and tools, and explores their usability with a visual modeler. Chapter 5 describes the Visual Modeler architecture and implementation. Chapter 6 presents case studies of integration of a VM with existing WfMSs. Finally, Chapter 7 presents conclusions and recommendations for future development.

Chapter 2

A SURVEY OF EXISTING WfMSs

There are several commercial and open source WfMSs available. In this thesis, the following systems are reviewed: JBoss jBpm, Zebra, YAWL, Enhydra JaWE, XFlow, and WFGenSystem-UNF. These systems provide business modeling functionality. Some of the systems are intended for developers rather than for end users. Few of these systems have a graphical user interface (GUI) for creating and editing workflows, but none of these systems use Business Process Modeling Notation (BPMN) as a diagramming notation.

2.1 JBoss jBPM

JBoss jBPM is an open source WfMS that can be used as a standalone application or can be embedded within a Java application (standard or enterprise edition) [jBPM]. JBoss jBPM provides a process definition language called jPDL for defining process workflows in Java [jPDL]. Also included, as part of the package, is the jBPM graphical

designer, which is currently available as an Eclipse plug-in (see Figure 2). The graphical designer is not based on BPMN. JBoss jBPM is available for most operating systems.

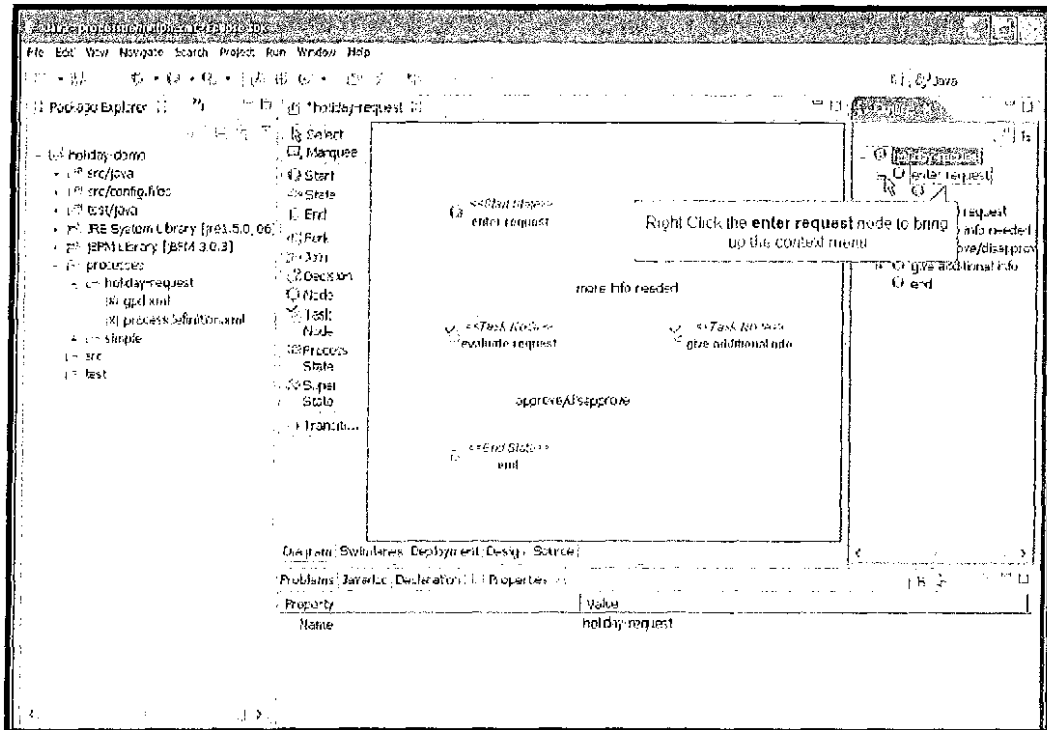


Figure 2: JBoss jBPM Graphical Process Designer Eclipse Plug-in

2.2 Zebra

Zebra is an open source workflow engine developed in Java [Zebra]. It includes a GUI designer developed in Visual Basic. The graphical designer is not based on BPMN. Zebra is available for most operating systems, provides

persistence through Hibernate, and has an interface for an XML loader. Figure 3 shows a screen shot of this system.

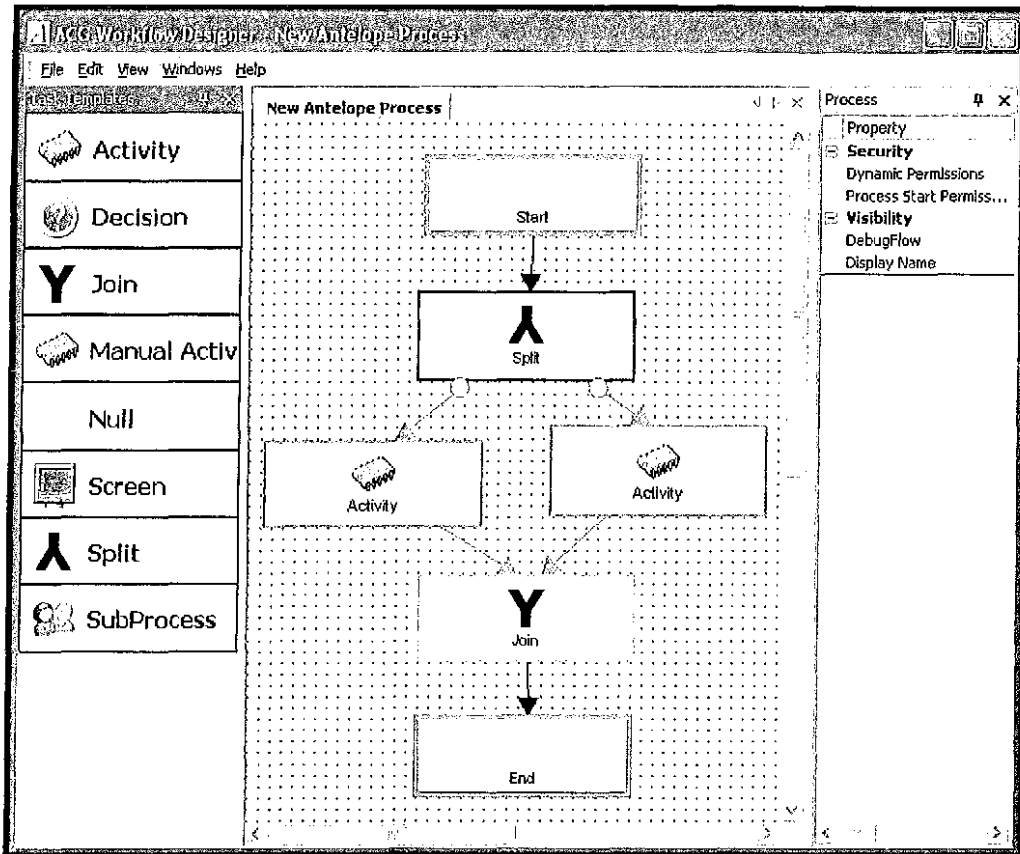


Figure 3: Zebra GUI Designer

2.3 YAWL

Yet Another Workflow Language (YAWL) is a workflow/BPM system [YAWL]. YAWL consists of a workflow engine and a visual editor. It is developed in Java. YAWL uses an XML schema, XQuery, XPath, and XForms for data storage and

2.4 XFlow

XFlow is an open source J2EE-based process management system that provides a platform for building, executing, and managing business processes and workflows [XFlow]. XFlow runs within a servlet container and comes with JBoss server implementation. XFlow does not provide any graphical workflow modeler or process monitoring capabilities.

2.5 Enhydra JaWE

Enhydra Java Workflow Editor (JaWE) is a workflow process editor based on the WfMC's XPDL specifications for the workflow file format [JaWE]. JaWE uses Enhydra Shark as the workflow engine, also based on XPDL specifications, to make a complete WfMS. XPDL stands for XML Process Definition Language used to represent the Workflow Process Diagram [XPDL]. XPDL provides the means to save and exchange the process diagram among applications. JaWE XPDL editor can be used with any notation-based diagram that can generate XPDL format files. The workflow editor is not based on any standard business process notation (see Figure 5).

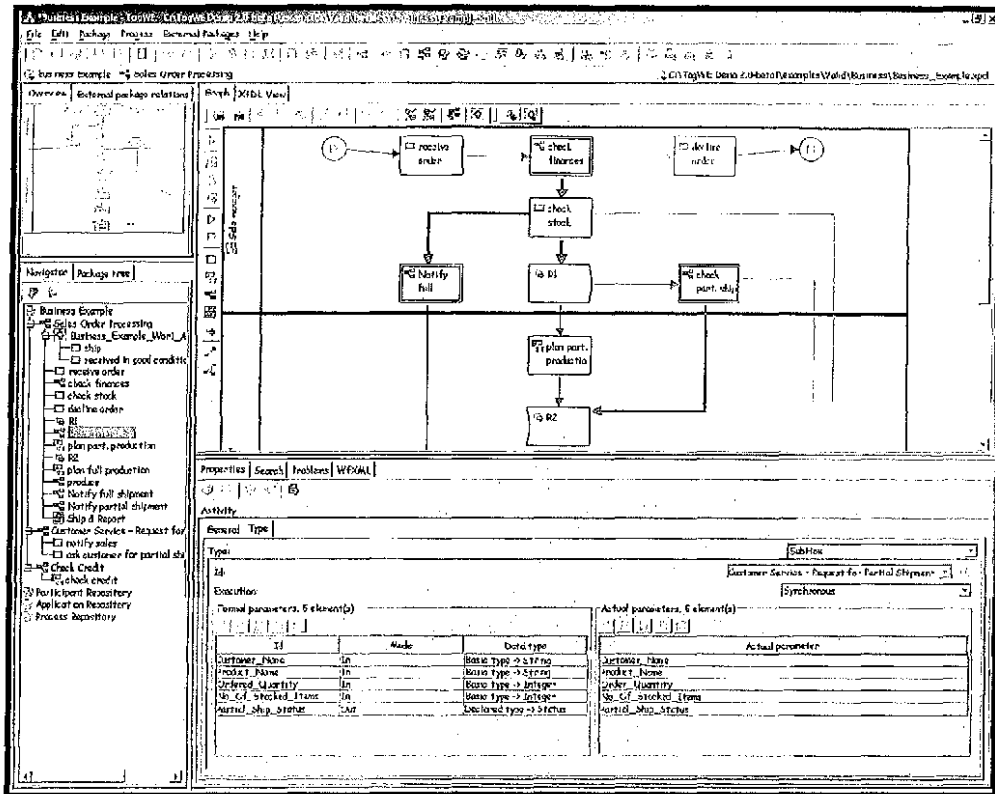


Figure 5: Enhydra JaWE in Together Professional

2.6 WGenSystem-UNF

WGenSystem-UNF is a web application, developed by Stacy Hutchings as part of her MS project. It generates customizable web-based document WfMSS [Hutchings05]. The motivation for and focus of WGenSystem-UNF are the needs of the end user. The system's workflow engineer interface provides functionality to define a form, a workflow, and a process definition. Figure 6 shows the current style

WGenSystem-UNF offers to final users, which is form-driven.

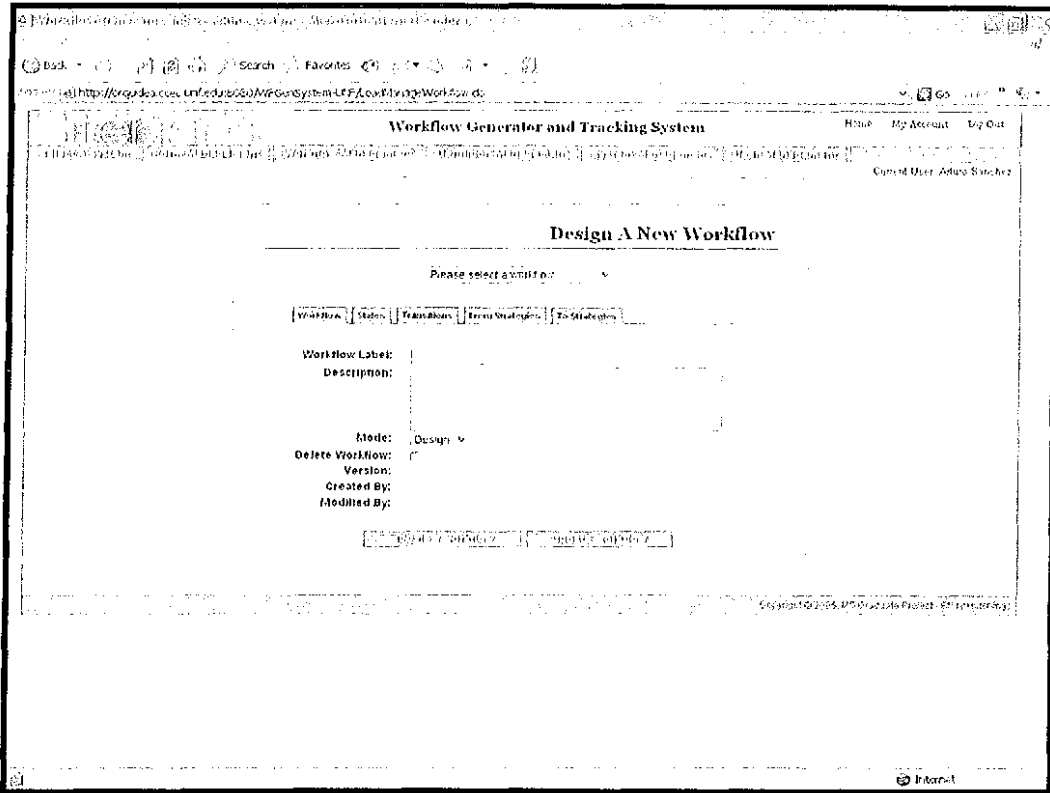


Figure 6: Interaction Style Currently Used by WGenSystem-UNF to Design, Update and Modify a Workflow

Either the systems with embedded visual modelers, discussed in the previous sections, do not provide easy integration with existing applications, do not use a standard notation, or do not provide clearly defined interfaces so they can be used as a standalone application with full menu options to manipulate diagrams. Appendix A summarizes the advantages and disadvantages of the various

systems and tools. Table A1 shows that not every system studied provides graphical designing capability. None of the systems with a graphical designer have graphical designer user interface customization flexibility. None of these systems use BPMN for workflow. Graphical designer of these systems cannot be integrated with other WfMSs. The only exception is Enhydra JaWE, which has graphical designer that can only be integrated with desktop-based WfMSs.

The VM developed as part of this work solves all of the above problems. It provides a visual interface to define workflows using a standard notation, and offers features such as editing, saving, and updating of workflows. The VM can easily be integrated with any WfMS application through well-defined interfaces (see Appendix A).

Chapter 3

A SURVEY OF MODELING NOTATIONS

The Unified Modeling Language's (UML) 2.0 Activity Diagrams (AD) and BPMN provide notations for graphically specifying workflow-based business processes [Bock03]. In this thesis UML 2.0 ADs and BPMN are reviewed and compared, from the perspective of some of the workflow patterns proposed by van der Aalst et al. [van der Aalst03], as shown in Appendix B.

The BPMN, developed by BPM Initiative (BPMI) and now maintained by Object Management Group (OMG)², provides a graphical notation for expressing business processes in a business process diagram [OMG]. The objective of BPMN is to support BPM by both technical users and business users, by providing them with a notation that is intuitive to business users, yet able to represent the semantic of complex processes. The BPMN specification also provides a mapping from the visual elements of the notation to the underlying constructs of execution languages, particularly

² The OMG is an international, open membership, not-for-profit computer industry consortium.

the Business Process Execution Language for Web Services (BPEL4WS)³ [Andrews03]. The introduction of BPMN creates a bridge between the business and IT layers of an organization. As Stephen A. White describes:

The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes [White04A].

UML is a language with a very broad scope that covers a large and diverse set of application domains. Not all of its modeling capabilities are necessarily useful in all domains or applications. The modeling concepts of UML are grouped into language units. ADs are intended for modeling computational and business/organizational processes.

UML 2.0 AD and BPMN share many of the same shapes for the same purposes. There are some differences among the visual elements, with BPMN containing fewer core objects and having variations on these objects to handle the complexities that might arise in modeling processes. As

³ BPEL4WS, provides a language for the formal specification of business processes and business interaction protocols.

shown in Table B3 (see Appendix B), BPMN offers separate notations for displaying variations of flow. BPMN uses notations consistently, for example, a diamond for a decision and a rectangle for activity. UML 2.0 AD and BPMN notations provide similar representations for most of the control-flow patterns. The only exception is ADs do not have an adequate graphical representation of the Interleaved Parallel Routing pattern and the Synchronizing Merge pattern.

BPMN and UML 2.0 AD can be used as a graphical front-end, to capture BPEL process descriptions. BPEL is an XML-based language for the definition of business processes. BEA, IBM, Siebel Systems, and Microsoft developed the first version of BPEL. It was named BPEL4WS [Andrews03]. BPEL4WS 1.1 was submitted to the Organization for the Advancement of Structured Information Standards (OASIS) for standardization purposes [OASIS]. The OASIS technical committee came up with the name Web Services BPEL, version 2.0 in September 2004. This led to a broader acceptance of BPEL in industry and it emerged as a standard to address business process integration needs. BPEL is a language that allows business analysts to define business processes consistently, within or across companies. Because BPEL is

based on XML standards, it ensures interoperability when integrating with other systems. The OASIS technical committee did not suggest any graphical notation for Web Services BPEL. BPMN can be used as a graphical front-end, to capture BPEL process descriptions. White described a translation from BPMN to BPEL [White05]. An approach for the transformation of AD and BPMN, in the context of BPEL, is described by Kalnins [Kalnins06], while Ouyang et al. came up with a more detailed translation and developed a tool for translating BPMN to BPEL [Ouyang06].

"The two diagrams [BPMN and UML 2.0 AD] also share the characteristic of being a view (a diagram) for the Business Process Definition metamodel being developed through an RFP [request for proposal] process in the OMG" [White04B]. Wohed et al. conducted a study for pattern-based analysis of BPMN and UML 2.0 AD and concluded that BPMN and UML 2.0 AD are almost fully overlapping [Wohed05].

In 2001, the BPMN effort started to create a notation to draw business process diagrams for business people to use. In 1994, the UML effort started to standardize modeling for software development for technical people. The ADs

were included subsequently, in an effort to refine UML for business people, but it is still more technically oriented. The membership of the BPMI Notation Working Group represents a large segment of the BPM community. They have come to a consensus that BPMN should serve as the standard. For these reasons, BPMN was chosen as the notation for the VM.

Chapter 4

DIAGRAMMING TOOLS AND FRAMEWORKS

The Visual Modeler that was developed as part of this thesis was built using a diagramming framework. In this chapter, we present a review of some frameworks that were considered.

4.1 JGraphPad Pro

JGraphPad Pro is a commercial tool used to create diagrams such as graphs, database layouts, and business process workflows using BPMN. It uses the JGraph and JGraphLayout packages, which are open source frameworks, and allows easy integration with existing applications. It provides customization of icons and configuration of the application, using XML. JGraphPad Pro offers additional features needed for the integration of the VM with other components. However, it is priced at \$1,495, which prevented us from purchasing it for this work.

4.2 ILOG JViews Diagrammer

ILOG JViews Diagrammer provides a set of configurable Java components for creating graphical editing, visualization, supervision, and monitoring tools. The Diagrammer includes a customizable BPMN modeler that can be used with desktop or web applications. The developer license for Diagrammer is priced at \$4,000, which also was not within the budget for this work.

4.3 Business Process Visual Architect

Business Process Visual Architect (BP-VA) is a standalone BPMN-based visual diagramming tool. BP-VA is a desktop-based tool that provides limited customization options for integration. BP-VA cannot run in a browser and cannot be integrated with an existing web-based application, as it requires full installation on a client's machine with a valid license key for execution. BP-VA is priced at \$99 and is part of the Visual Paradigm Suite priced at \$1,999.

4.4 Enterprise Architect

Enterprise Architect is a UML 2.0-based modeling tool. It is a full featured desktop application, but does not provide support for BPMN, for process flow diagrams. The purpose of this research work was to build a VM, which supported BPMN; therefore, the UML notation-based tool was not the right fit. However, even with the current configuration of this tool, which does provide support for BPMN, it could not be integrated because it requires complete installation on the client's machine, with a separate license for its use. This tool is priced at \$135.

4.5 Enhydra JaWE

Enhydra JaWE is the first open source graphical workflow process editor based on the WfMC XPDL specification [XPDL]. Professional and community editions of this tool are available. Both the editions are available as plug-ins for Borland Together Software [BT] and the commercial product is called Together Workflow Editor [TWE]. The professional edition provides more features than the community edition. The editor can only be used as part of

the Together Workflow Editor software, thus was not the right tool for this work. The community edition is free and the professional edition of Together Workflow Editor is available for \$699.

4.6 Java Swing and Java 2D Frameworks

Java Swing [SWING] and Java 2D [JAVA2D] graphical packages provide a rich set of libraries to build GUIs, graphs, and diagrams. The business process flow diagrams created using these set of libraries do not provide any interactive features. To make a powerful VM requires a layout engine, user interface, and an adapter. Though not impossible, creating a layout engine from scratch would be an extensive task, thus was outside of the scope of this project, due to time constraints. The framework is free to download, but was not used for the project, because of the need to create a layout engine.

4.7 JGraph and JGraphLayout Frameworks

JGraph provides a very extensive set of packages to custom build a business process-diagramming tool. It provides a swing-like GUI package for the interface and a layout

package, providing automatic positioning functionality and ease of use. These features enable rapid application development and deployment. JGraph and JGraphLayout are available for \$495 for commercial use and are free for non-commercial use.

Of the tools and frameworks evaluated (see Appendix A), the JGraph and JGraphLayout frameworks seemed to be the best fit for this work, due to following reasons. These frameworks provide support for BPMN notation, rich set of application programming interfaces (API) for user interface, powerful layout engine that promised to reduce development time, flexibility to integrate with web-based applications, and API for creating custom database or XML adapter. JGraph and JGraphLayout applications cannot run inside a browser, but can be integrated with web applications, using the Web Start technology or as an Applet. The web version of JGraph and JGraphLayout, called MxGraph, is currently available as a beta version only [MXGRAPH]. MxGraph is capable of providing interactive workflow diagrams inside a browser. This would have been the right framework for the study, but because it was not available for our purposes at that time, we decided upon the JGraph and JGraphLayout frameworks.

Chapter 5

THE VM: SOFTWARE ARCHITECTURE AND IMPLEMENTATION

The VM is a standalone Java application for designing BPMN-based workflows. The VM is based on open source packages, JGraph and JGraphLayout, the Java 1.5 Swing package, and Java 1.5 SDK. JGraph provides the ability to easily create workflow components like nodes, ports, and the interconnectivity mechanism, while JGraphLayout provides the implementation of layout algorithms and graph resizing capabilities, making the task of designing complex workflows simpler. The JGraph package API provides ways to persist data, in the form of XML documents or directly to a database. The VM built as part of this thesis uses an XML document for storing the workflow definition.

The VM can design workflows using interactive user input or modified existing workflows, by taking input in the form of an XML file validated against the JGraph XML Schema (see Appendix C). The designed workflows are

persisted in the form of an XML file conformant to a JGraph XML Schema.

The architecture of the VM built as part of this thesis aims at enabling an easy integration of this tool with existing WfMS. Two major components define The VM's architecture: the VM user interface and the connectivity module. The user interface provides designing capabilities, whereas the connectivity interface provides a communication channel between the user application and the VM's user interface. Figure 7 provides an overview of the architecture.

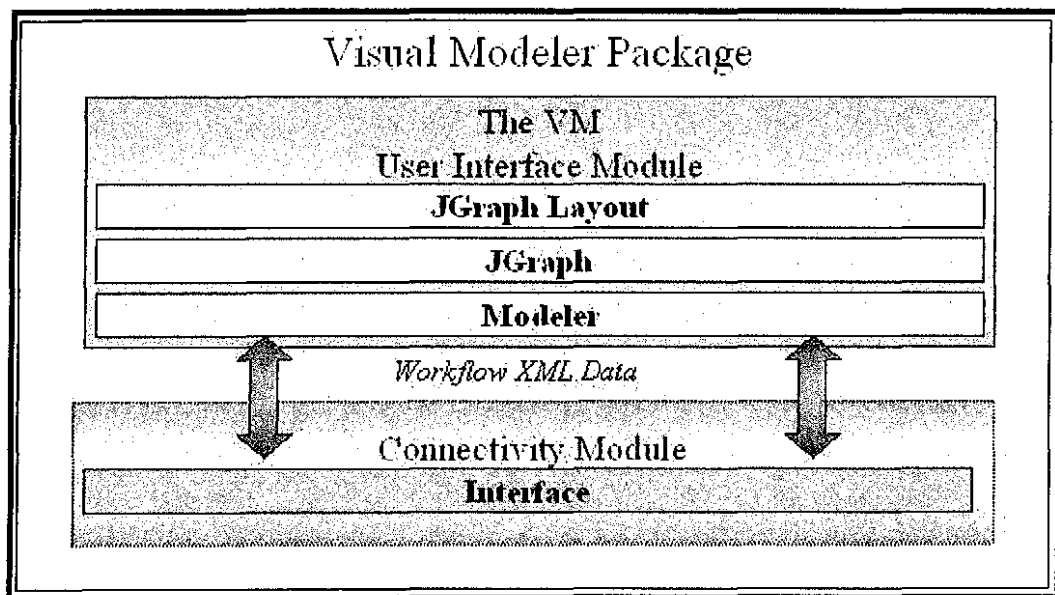


Figure 7: VM Architecture

The VM application (user interface module) consists of two packages, as described in Table 1.

Package	Description
edu.unf.soc.vm.ui	Package which consists of the base modeler and all User Interface (UI) components of the BPMN used in the workflow design
edu.unf.soc.vm	Package which consists of the extension of the base modeler, to provide XML persistence of the workflow and sub-menus for the workflow object properties

Table 1: User Interface Packages

The user interface provides a limited set of BPMN notation that can be used in the workflow design. Figure 8 shows the toolbar available for workflow designing and editing, including the set of BPMN entities.



Figure 8: VM Toolbar

The toolbar buttons available are the following, from left to right: Save, Start, Process, OR-Split/OR-Merge, AND-Split/AND-Merge, Deferred Choice, End, Toggle Connect Mode, Undo, Redo, Bring to Front, Send to Back, Reset

Zoom, Zoom In, Zoom Out, Group, Ungroup, Collapse, Expand, and Expand All.

Using the rich user interface toolbar, a simple workflow can be designed in few mouse clicks. As workflow objects are created, unique default labels are assigned to them. The workflow in the Figure 9 shows two processes and three labels. The start and end states have the default labels, Start and End, which cannot be modified.

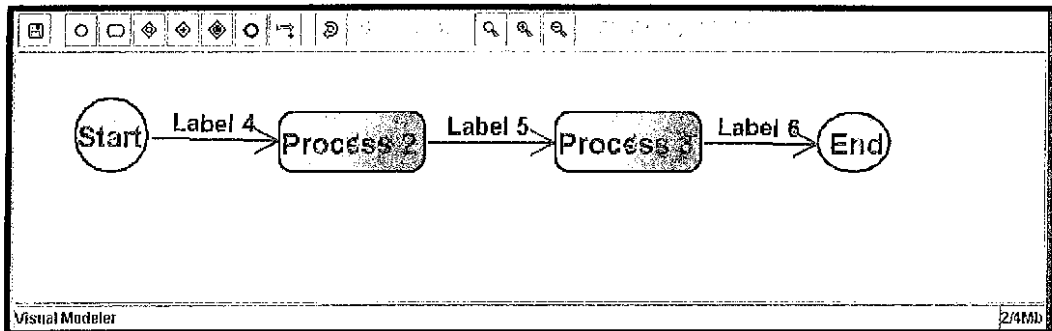


Figure 9: A Simple Workflow Using the Rich User Interface

The user interface also provides an option to edit individual properties of the workflow objects. Figure 10 shows sub-menu options for the selected process, Process 2. Figure 11 shows sub-menu options for the selected flow, Label 5. Figure 12 shows the sub-menu options for the workflow. The properties option can be customized by the user through the connectivity interface.

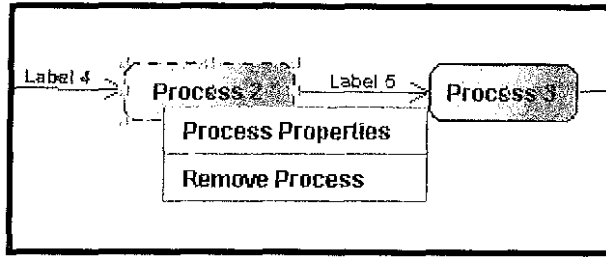


Figure 10: Process Sub-Menu

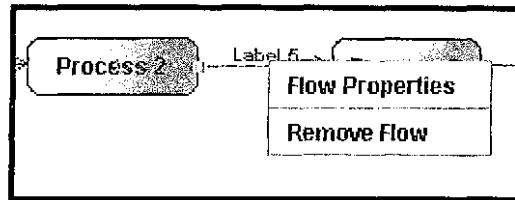


Figure 11: Flow Sub-Menu

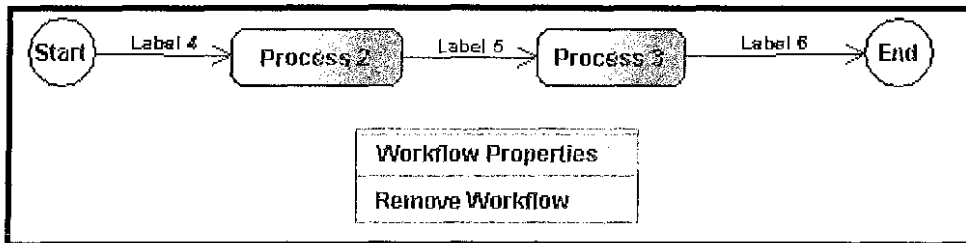


Figure 12: Workflow Sub-Menu

The connectivity interface module is provided for customization of the user interface and custom implementation of the VM. The module consists of a Java package, edu.unf.soc.vm.connectivity, with one interface class that can be implemented, as required. The methods in the interface class are listed in Table 2.

Interface edu.unf.soc.vm.connectivity.ModelerConnection:

Method	Description
displayWorkflowProperties()	Method to display workflow properties, invoked from the workflow sub-menu of VM
displayProcessProperties()	Method to display process properties, invoked from the process sub-menu of VM
displayFlowProperties()	Method to display flow properties, invoked from the flow sub-menu of VM
getNewWorkflowId()	Method to return the workflow id when a new workflow is created
readWorkflow()	Method to read the saved XML representation of the workflow
removeWorkflow()	Method to remove the workflow and its related information
removeProcess()	Method to remove the process and its related information
removeFlow()	Method to remove the flow and its related information
saveWorkflow()	Method to save the XML representation of the workflow and related information
saveProcess()	Method to save the process information
saveFlow()	Method to save the flow information

Table 2: Available Methods of Interface Class

Together the VM user interface and connectivity interface make the VM customizable. They can be added to any existing workflow application to enhance the workflow

design capabilities. The next chapter details the case studies for two applications, showing how the required interfaces were customized.

Chapter 6

CASE STUDIES

Figure 13 shows how the same VM was integrated with two different types of applications, using the same user interface and custom connectivity. The WFGenSystem-UNF application uses database persistence and the XFlow application uses file system persistence.

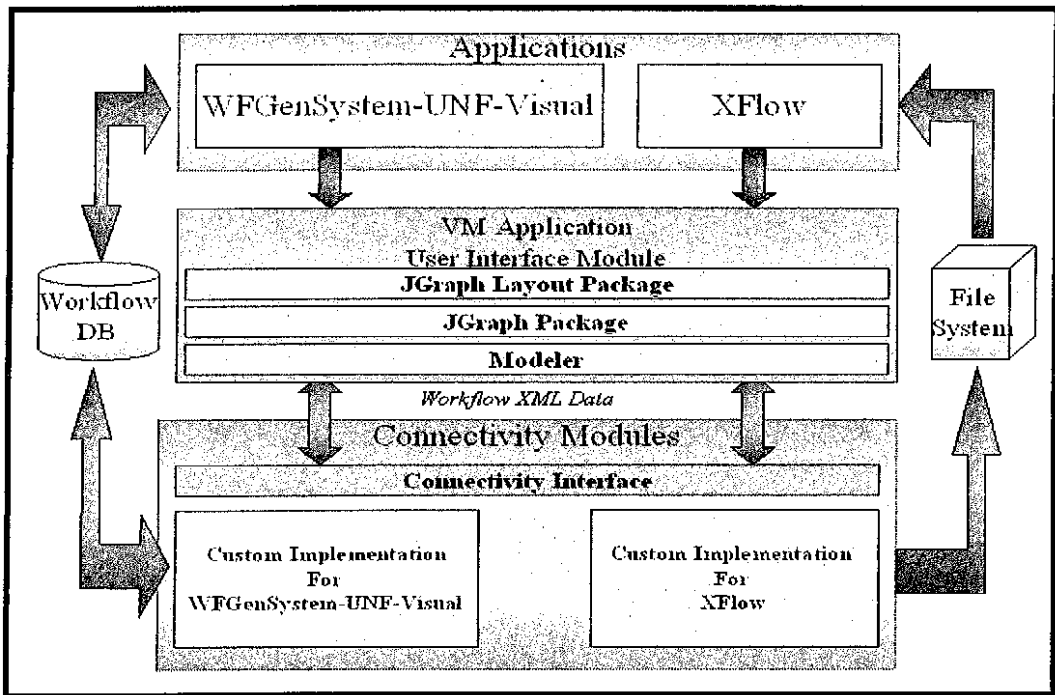


Figure 13: VM Architecture with Respect to Integration with WFGenSystem-UNF and XFlow Applications

6.1 Case Study 1: WFGenSystem-UNF

WFGenSystem-UNF is a web-based document workflow management system. WFGenSystem-UNF has been implemented, using the model view controller architectural pattern. It uses Java server pages for the user interface, MySQL for database persistence, and consists of multiple modules. The modules, categorized based on the user group, are Administrator, Workflow Engineer, and Workflow User. The Administrator module provides user and system management. The Workflow Engineer module provides form, workflow, and process management. The Workflow User module provides task management. This case study focused on the workflow management of the Workflow Engineer module. Figure 14 and Figure 15 demonstrate the VM integration with WFGenSystem-UNF, in two levels of details.

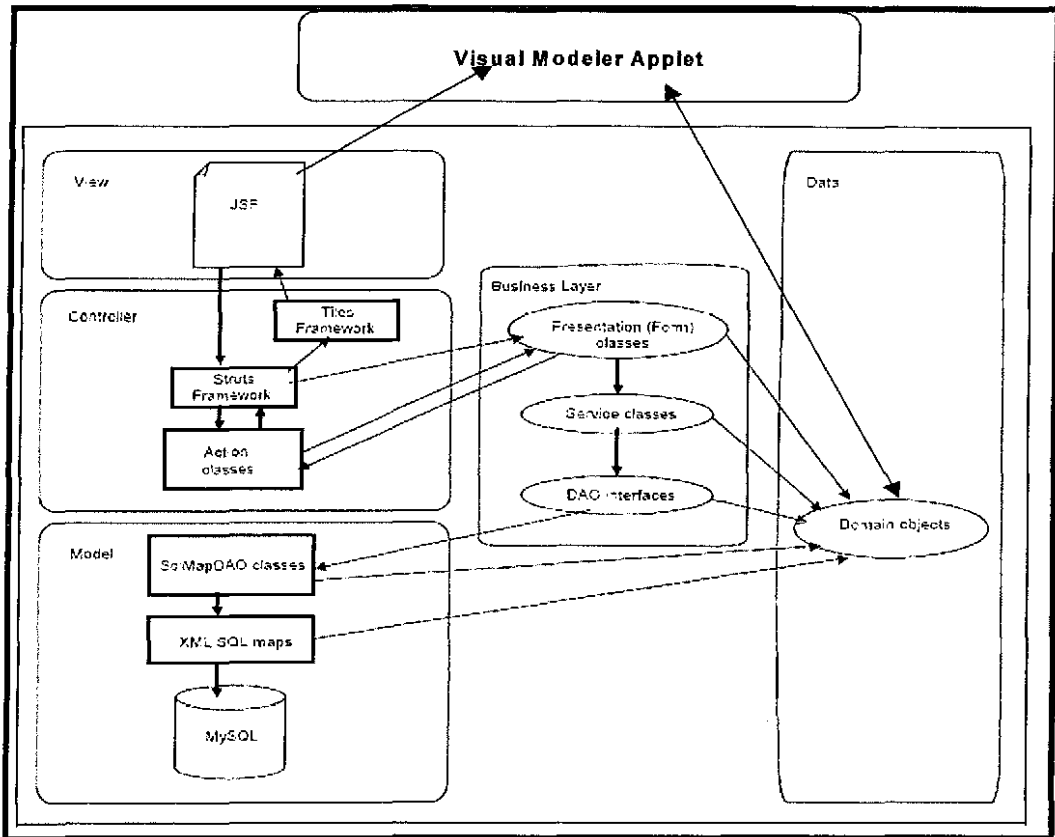


Figure 14: WGenSystem-UNF Architecture Diagram with the Integration of VM

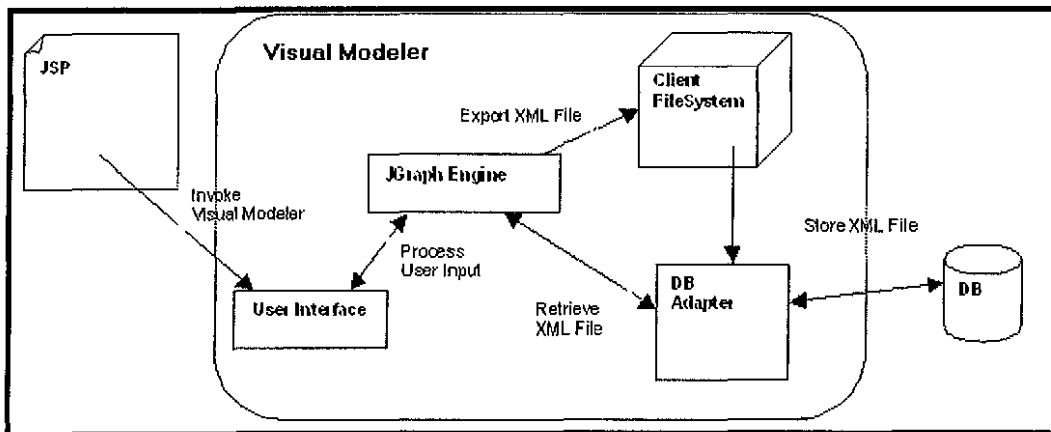


Figure 15: Details of the VM Integration with WGenSystem-UNF

The VM was integrated with WGenSystem-UNF in the form of an embedded applet. Clicking on New or Edit Workflow from the Design Workflow menu option of the Workflow Management tab starts the VM (see Figure 16).

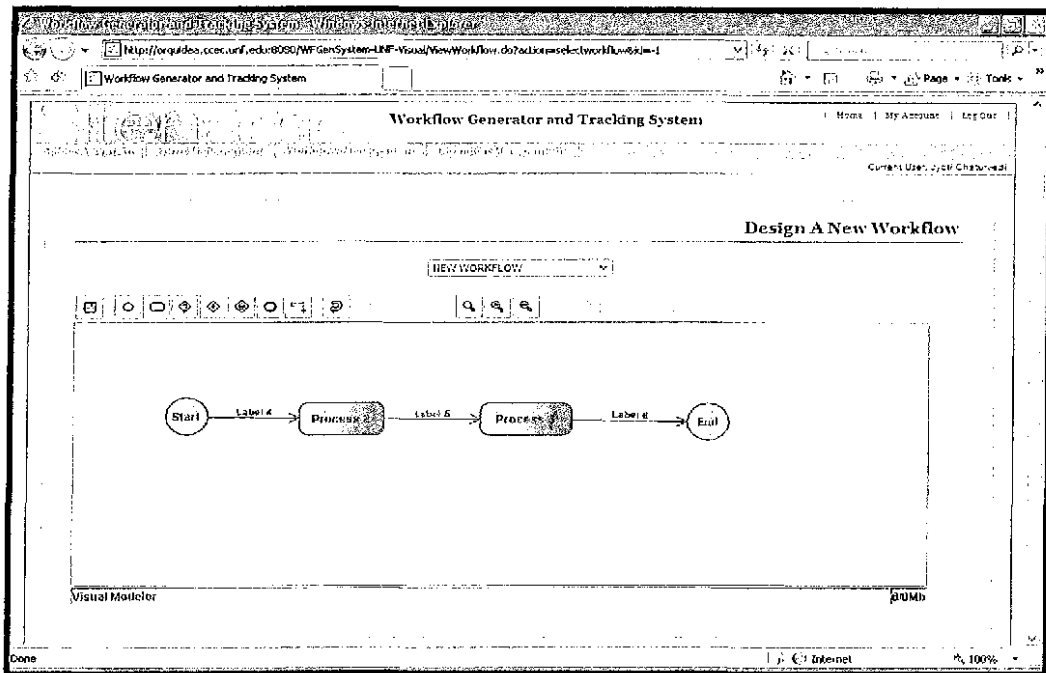


Figure 16: VM Integrated with WGenSystem-UNF

The customization to integrate the VM with WGenSystem-UNF was done by implementing an instance of the connectivity module to target this specific integration. The packages and their methods are described in Tables 3-5.

Package edu.unf.soc.vm.connectivity:

Class	Description
ModelerConnection	Interface definition with the available methods
ModelerConnectionFactory	Factory class to get the instance of the ModelerConnection implementation
ModelerConnectionImpl	Implementation class that implements the interface methods
ModelerConnectionHelper	Helper class with methods used in the ModelerConnectionImpl

Table 3: Classes of Package edu.unf.soc.vm.connectivity

Package edu.unf.soc.vm.connectivity.db:

Class	Description
Workflow	Class to represent the workflow database entity used by WFGenSystem-UNE for persistence of workflow data
WorkflowVersion	Class to represent the workflowversion database entity
Process	Class to represent the state database entity
Flow	Class to represent the transition database entity
DeferredChoice	Class to represent the deferredchoice database entity

Table 4: Classes of Package edu.unf.soc.vm.connectivity.db

Package edu.unf.soc.vm.connectivity.dialog:

Class	Description
WorkflowDialog	Class to create the Workflow properties dialog
ProcessDialog	Class to create the Process properties dialog
FlowDialog	class to create the Flow properties dialog

Table 5: Classes of Package
edu.unf.soc.vm.connectivity.dialog

Using the VM, a new workflow in the WGenSystem-UNF can be designed following these steps:

- 1 From the Workflow Management menu in WGenSystem-UNF, choose the Design Workflow option.
- 2 Select the New Workflow option from the selection list, to design a new workflow.
- 3 The VM applet starts with the Workflow Properties dialog (see Figure 17). Enter the Label, Description, and the Mode, then click Save.

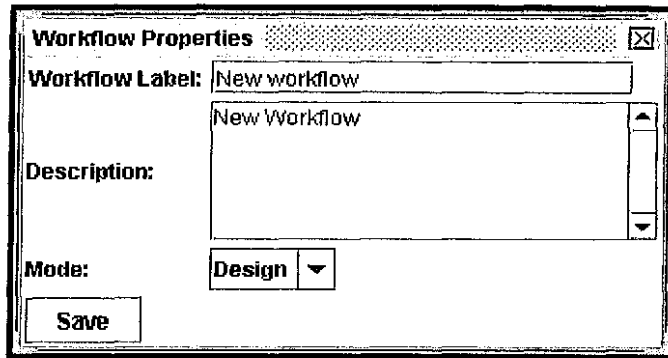


Figure 17: Workflow Properties Dialog

- 4 Using the toolbar in the modeler, design the workflow.
- 5 Edit the Process Label and Flow Label, and properties, using the corresponding dialog boxes (see Figures 18 and 19).

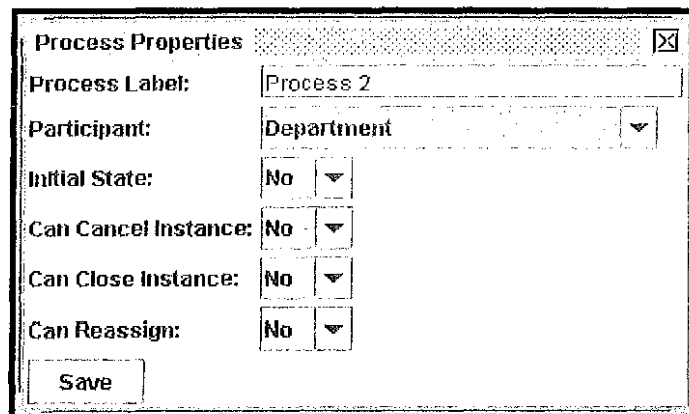


Figure 18: Process Properties Dialog

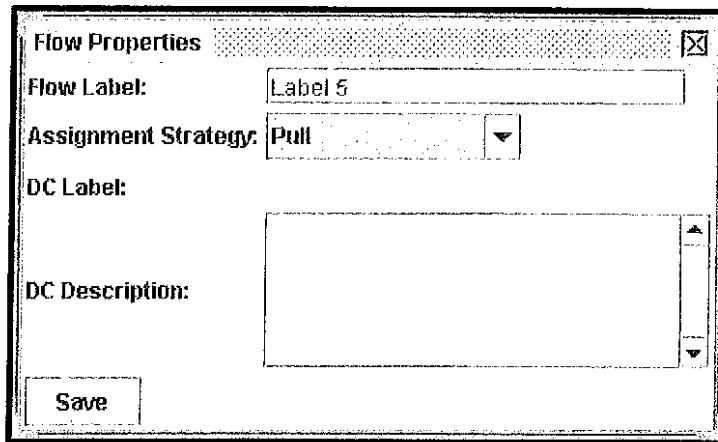


Figure 19: Flow Properties Dialog

- 6 Use the Save option in the toolbar to create the individual workflow States, Transitions, and Workflow Definitions.
- 7 Publish the Workflow for use by the Definition Management module.

Use of the VM in this application seemed to make the visualization of a complex workflow easier and faster. Appendix D presents a demonstration of the WFGenSystem-UNF-Visual application, using the VM from the design of the workflow network to the implementation of a complete system.

6.2 Case Study 2: XFlow

XFlow is a JEE-based process management system. XFlow runs in an EJB and Servlet container, using JBoss 4.0 and Tomcat for container implementation. XFlow does not have any GUI for workflow design or workflow process monitoring. The VM can easily be integrated with XFlow.

XFlow1.2.1, used in this study, is distributed in the form of a compressed zip file. The XFlow package comes with the JBoss 4.0 (bundled with Tomcat) server and some workflow examples that can be deployed and tested on the XFlow server. The XFlow package does not provide any interface to design workflows. Currently, users define their workflows using a text editor to create an XML file that validates to the XFLOW file schema. For this case study, the VM connectivity module was implemented for the XFlow application, so it generates the XML files, which conform to the XML schema for XFlow.

The XFlow server can be deployed and run from a MS Windows or UNIX shell. For this case study, the Windows version of XFlow was used, running on the Windows XP Professional operating system and the Java 1.4.2_15 SDK.

The VM application is launched as an Applet, using the Java Applet Viewer, as shown in Figure 20. Once the VM starts, a workflow can be designed.

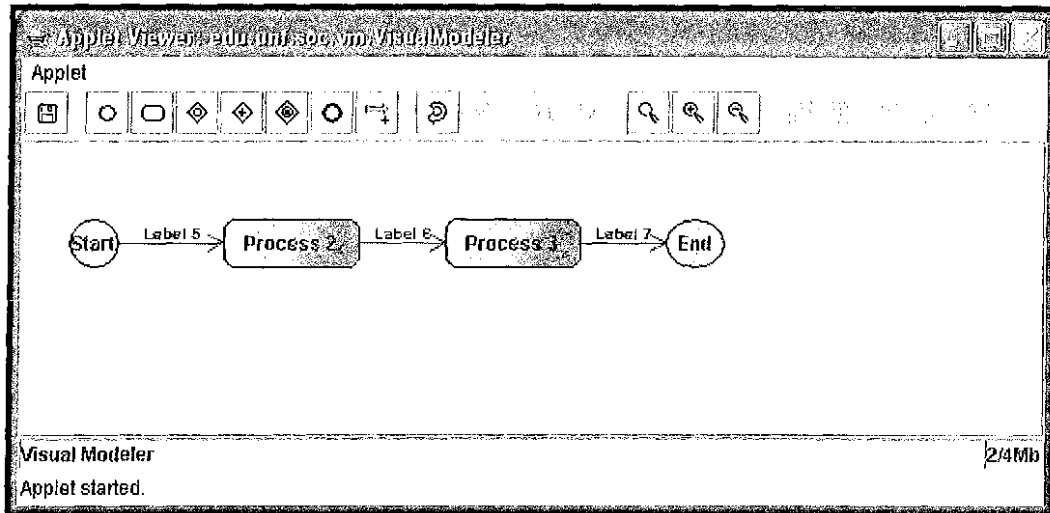


Figure 20: VM Applet Integrated with XFlow

To generate the XML output (<file>.xflow), the connectivity module was implemented, using the packages identified in Tables 6-8.

Package edu.unf.soc.vm.connectivity:

Class	Description
ModelerConnection	Interface definition with the available methods
ModelerConnectionFactory	Factory class to get the instance of the ModelerConnection implementation
ModelerConnectionImpl	Implementation class that implements the interface methods
ModelerConnectionHelper	Helper class with methods used in the ModelerConnectionImpl

Table 6: Classes of Package edu.unf.soc.vm.connectivity

Package edu.unf.soc.vm.connectivity.db:

Class	Description
Workflow	Class to represent the workflow properties

Table 7: Classes of Package edu.unf.soc.vm.connectivity.db

Package edu.unf.soc.vm.connectivity.dialog:

Class	Description
WorkflowDialog	Class to create the Workflow properties dialog

Table 8: Classes of Package
edu.unf.soc.vm.connectivity.dialog

The interface methods implemented for the XFlow integration are identified in Table 9:

Method	Description
<code>readWorkflow()</code>	Method to get the existing workflow XML file
<code>saveWorkflow()</code>	Method to save the workflow XML file and create input file to XFlow (<file>.xflow)
<code>displayWorkflowProperties()</code>	Method to display workflow properties dialog (see Figure 21)

Table 9: Methods Implemented for XFlow

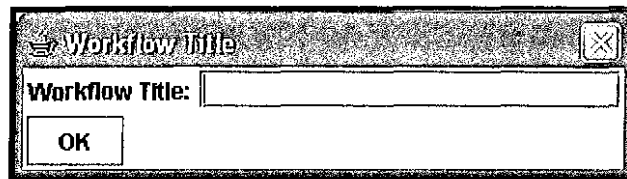


Figure 21: Workflow Properties Dialog for XFlow

The VM generates the output XFLOW file, as shown in Figure 22. Appendix E presents a demonstration of the XFlow application, using the VM to design a workflow.

The VM can be integrated with virtually any application, web-based or desktop. This is possible if a correct implementation of the interfaces in the connectivity module is provided.

```
<xflow name="SimpleWorkflow">
  <nodes>
    <node id="StartNode" type="Start"/>
    <node id="P1" type="Process"/>
    <node id="P2" type="Process"/>
    <node id="P3" type="Process"/>
    <node id="P4" type="Process"/>
    <node id="P5" type="Process"/>
    <node id="EndNode" type="End"/>
  </nodes>
  <transitions>
    <transition from="StartNode" to="P1"/>
    <transition from="P1" to="P2"/>
    <transition from="P2" to="P3">
      <rule>[intValue < 10]</rule>
    </transition>
    <transition from="P2" to="P4">
      <rule>[intValue >= 10]</rule>
    </transition>
    <transition from="P3" to="P5"/>
    <transition from="P4" to="P5"/>
    <transition from="P5" to="EndNode"/>
  </transitions>
</xflow>
```

Figure 22: XFLOW File Output from VM

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

A wide variety of WfMSs were reviewed. Not all of the systems reviewed were created based on the needs of the end user. Few WfMSs provide a GUI for defining workflows. The goal of this work was to enhance existing WfMSs, by providing a Visual Modeler for visually defining workflows, using BPMN.

The VM was designed to empower end users to define complex workflow diagrams easily. While one of the applications used as a test case (WFGenSystem-UNF) allows end-users to define one component at a time for the workflow (i.e., states, transitions, and rules), the VM allows users to define these components as a whole, giving users a high-level perspective of the workflow design. XFlow, the second application tested, natively uses an XML editor for creating workflows. In this case, the VM provided for a visual representation of the workflow design.

A complex and powerful WfMS may remain under utilized, when workflows are defined using a non-friendly interface. The VM enhances existing WfMSs, as demonstrated in the case studies, making them more manageable for a wider community of users.

The VM was designed so the integration with existing systems can be approached in a systematic and clear manner, by implementing well-defined interfaces. This was also demonstrated in the case studies.

7.2 Future Work

The current VM implements a subset of the BPMN, namely: - Process, Start and End Event, OR-Split, OR-Merge, AND-Split, AND-Merge, and Deferred Choice. Implementation of the whole BPMN set would enhance the VM modeling capabilities.

It is possible to add an option, which would allow users to design their workflows using a grid. This would make alignment and placement of workflow objects easier and more precise, resulting in a more organized layout.

REFERENCES

Print Publications:

[Bock03]

Bock, C., "UML 2 Activity and Action Models," Journal of Object Technology 2, 4, (July-August, 2003), pp. 43-53.

[Hutchings05]

Hutchings, S., "An End-User Development Approach to Building Customizable, Web-Based, Document Workflow Management Systems," MS Project, Department of Computer and Information Sciences, University of North Florida, Jacksonville, 2005.

[Kalnins06]

Kalnins, V., "Use of UML and Model Transformations for Workflow Process Definitions," Communications of the 7th International Baltic Conference on Databases and Information Systems, Vilnius, Lithuania (July, 2006), pp. 3-14.

[Ouyang06]

Ouyang, C., W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede, "Translating BPMN to BPEL," BPM Center Report, BPM-06-02 (2006).

[van der Aalst03]

van der Aalst, W. M. P., A. H. M. ter Hofstede, B. Kiepuszewski, and A. Barros, "Workflow Patterns," Distributed and Parallel Databases 14, 3, (July, 2003), pp. 5-51.

[WfMC95]

Workflow Management Coalition (WfMC), "The Workflow Reference Model," TC00-1003 (January, 1995).

[WfMC99]

Workflow Management Coalition (WfMC), "Terminology & Glossary," TC-1011, 3 (February, 1999), pp. 8.

[White04A]

White, S. A., "Introduction to BPMN," BPTrends, (July, 2004), pp. 1.

[White04B]

White, S. A., "Process Modeling Notations and Workflow Patterns," BPTrends, (March, 2004), p. 24.

[White05]

White, S. A., "Using BPMN to Model a BPEL Process," BPTrends, (March, 2005).

[Wohed05]

Wohed, P., W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell, "Pattern-based Analysis of BPMN - An extensive evaluation of the Control-flow, the Data and the Resource Perspectives," BPM Center Report, BPM-05-26 (2005).

Electronic Sources:

[Andrews03]

Andrews, T., "Business Process Execution Language for Web Services," BEA Systems, (May, 2003),
<ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>, last accessed October 27, 2007.

[BT]

Borland Together,
<http://www.borland.com/us/products/together/index.html>, last accessed October 27, 2007.

[JaWE]

Enhydra Java Workflow Editor,
<http://www.enhydra.org/workflow/shark/index.html>,
last accessed October 27, 2007.

[JAVA2D]

Java 2D API, Sun Microsystems,
<http://java.sun.com/products/java-media/2D/>, last
accessed October 27, 2007.

[jBPM]

JBoss jBPM, <http://www.jboss.com/products/jbpm>, last accessed October 27, 2007.

[JGraph]

Java Graph Visualization and Layout,
<http://www.jgraph.com/>, last accessed October 27, 2007.

[jPDL]

jBPM Process Definition Language,
<http://docs.jboss.org/jbpm/v3/userguide/jpdl.html>,
last accessed October 27, 2007.

[MVC]

Java BluePrints - J2EE Patterns, Model-View-Controller,
<http://java.sun.com/blueprints/patterns/MVC-detailed.html>, last accessed October 27, 2007.

[MXGRAPH]

JGraph mxGraph,
<http://www.mxgraph.com/pages/en/index.html>,
last accessed October 27, 2007.

[OASIS]

OASIS Web Services Business Process Execution Language,
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel,
last accessed October 27, 2007.

[OMG]

Object Management Group, <http://www.omg.org/>, last Accessed October 27, 2007.

[STRUTS]

Struts, The Apache Software Foundation,
<http://struts.apache.org/>, last revised October 18, 2007, last accessed October 27, 2007.

[SWING]

Creating a GUI with JFC/Swing, The Java Tutorials,
<http://java.sun.com/docs/books/tutorial/uiswing/>,
last accessed October 27, 2007.

[TWE]

Together Workflow Editor,
<http://www.together.at/together/prod/twe/index.html>,
last accessed October 27, 2007.

[Web Start]

Java Web Start Technology,
<http://java.sun.com/products/javawebstart/>,
last accessed October 27, 2007.

[WfMOpen]

WfMOpen Project, <http://wfmopen.sourceforge.net/>, last
accessed October 27, 2007.

[XFlow]

XFLOW Process Management System,
<http://xflow.sourceforge.net/>, last accessed October
27, 2007.

[XPDL]

XML Process Definition Language,
<http://www.wfmc.org/standards/XPDL.htm>, last
accessed October 27, 2007.

[YAWL]

Yet Another Workflow Language,
<http://www.yawl-system.com/>, last accessed
October 27, 2007.

[Zebra]

Zebra Workflow, <http://zebra.berlios.de/>, last accessed
October 27, 2007.

Comparison of WfMSSs, Diagramming Tools and Frameworks with the VM						
	Graphical Workflow Designer	Graphical Designer User Interface Customization	BPMN for Workflow	Graphical Designer Integration with Desktop Based WfMSS	Graphical Designer Integration with Web Based WfMSS	Workflow XML Persistence
Existing WfMSSs:						
JBoss jBPM	X					X
Zebra	X					X
YAWL	X					X
XFlow						X
Enhydra JaWE	X			X		X
WFGenSystem-UNF						
Tools and Frameworks:						
JGraphPad Pro	X	X	X	X		X
ILOG Jviews Diagrammer	X	X	X	X		X
Business Process - Visual Architect	X		X	X		
Enterprise Architect	X			X		
Enhydra JaWE	X			X		X
Java Swing and Java 2D Frameworks*		X	X			
JGraph and JGraphLayout Frameworks*	X	X	X	X	X	X
Visual Modeler	X	X	X	X	X	X
* These frameworks only provide basic capabilities that can be used to build a workflow designer with above features.						

Table A1: Comparison of WfMSSs, Diagramming Tools and Frameworks

APPENDIX B

BPMN AND UML NOTATION COMPARISON

This appendix presents the results of comparing UML 2.0 ADs and BPMN, as visual notations that can be used by users to specify workflow-based business processes. Tables B1-B5 contain a summary of the visual elements used by each notation. Tables B6-B9 contain a comparison of these two notations from the perspective of how they support van der Aalst's workflow patterns [van der Aalst03].

Table B1: Comparison of BPMN and UML Notations








Comparison of Activity/State/Action & Control Nodes of BPMN and UML Notations					
BPMN 1.0			UML 2.0		
Element	Description	Notation	Element	Description	Notation
Event	An event is something that "happens" during the course of a business process. These events affect the flow of the process and usually have a cause (trigger) or an impact (result). There are three types of Events, based on when they affect the flow: Start, Intermediate, and End.		Action	An action element describes a basic process or transformation that occurs within a system. It is the basic functional unit within an Activity diagram. Actions can be thought of as children of activities.	
Start	Start Event indicates where a particular process will start.		Initial	The <i>initial</i> element is used by the Activity and State Machine diagrams	 Initial
Intermediate	Intermediate Events occur between a Start Event and an End Event. It will affect the flow of the process, but will not start or (directly) terminate the process.			Depicts an exit from the system but has no effect on other executing flows in the activity	 Flow Final
End	As the name implies, the End Event indicates where a process will end.		Final	Indicates the completion of an activity	 Final

Table B2: Comparison of BPMN and UML Notations


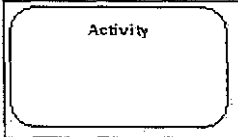

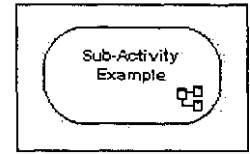






Comparison of Activity/State/Action & Control Nodes of BPMN and UML Notations					
BPMN 1.0			UML 2.0		
Element	Description	Notation	Element	Description	Notation
Process	A Task is an atomic activity that is included within a Process. A Task is used when the work in the Process is not broken down to a finer level of Process Model detail.		Activity	An activity organizes and specifies the participation of subordinate behaviors, such as sub-activities or actions, to reflect the control and data flow of a process	
Sub-Process	The details of the Sub-Process are not visible in the Diagram. A "plus" sign in the lower-center of the shape indicates that the activity is a Sub-Process and has a lower-level of detail.		Sub-Activity	A subactivity element is a pointer to a child Activity diagram	
Receive	BPMN uses Process notation with [Receive] label to indicate Receive action		Receive Activity	A receive element is used to define the acceptance or receipt of a request. Movement from a receive element occurs only once receipt is fulfilled according to its specification. The receive element comes in two	
Send	BPMN uses Process notation with [Send] label to indicate Send action		Send Activity	The send element is used to depict the action of sending a signal	
Group	The grouping can be used for documentation or analysis purposes.		Region	There are two types of regions supported. Expansion region AND Interruptible activity region	

Table B3: Comparison of BPMN and UML Notations

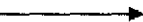
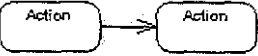

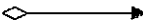

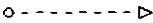
Comparison of Transition/Edges of BPMN and UML Notations					
BPMN 1.0			UML 2.0		
Element	Description	Notation	Element	Description	Notation
Normal Flow	Normal Sequence Flow refers to the flow that originates from a Start Event and continues through activities via alternative and parallel paths until it ends at an End Event.		Control Flow	The <i>control flow</i> is a connector linking two nodes in an Activity diagram. Control flow connectors bridge the flow between activity nodes, by directing the flow to the target node once the source node's activity is completed.	
Uncontrolled Flow	Uncontrolled flow refers to flow that is not affected by any conditions or does not pass through a Gateway.				
Conditional Flow	Sequence Flow can have condition expressions that are evaluated at runtime to determine whether or not the flow will be used. If the conditional flow is outgoing from an activity, then the Sequence Flow will have a mini-diamond at the beginning of the line.				
Default Flow	For Data-Based Exclusive Decisions or Inclusive Decisions, one type of flow is the Default condition flow. This flow will be used only if all the other outgoing conditional flow is not true at runtime.				
Message Flow	A Message Flow is used to show the flow of messages between two entities that are prepared to send and receive them.				

Table B4: Comparison of BPMN and UML Notations

Comparison of Transition/Edges of BPMN and UML Notations					
BPMN 1.0			UML 2.0		
Element	Description	Notation	Element	Description	Notation
Exception Flow	Exception Flow occurs outside the Normal Flow of the Process and is based upon an Intermediate Event that occurs during the performance of the Process.		Interrupt Flow	The <i>interrupt flow</i> is a toolbox element used to define the two UML concepts of connectors for Exception Handler and Interruptible Activity Region. An interrupt flow is also known as an activity edge.	
Compensation Association	Compensation Association occurs outside the Normal Flow of the Process and is based upon an event				
Data Object	Data Objects are considered Artifacts because they do not have any direct effect on the Sequence Flow or Message Flow of the Process, but they do provide information about what activities require to be performed and/or what they produce.		Object Flow	<i>Object flows</i> are used in Activity diagrams and State Machines. When used in an Activity diagram, an object flow connects two elements, with specific data passing through it.	
Decision	This Decision represents a branching point where Alternatives are based on conditional expressions contained within the outgoing Sequence Flow. Only one of the Alternatives will be chosen.		Decision	This node represents a point in an activity where a single incoming edge branches into several outgoing edges. You typically use constraints, also called guard conditions, on the outgoing edges to determine which edge should be followed. A decision may be shown by labeling multiple output transitions of an action with different guard conditions.	

Table B5: Comparison of BPMN and UML Notations

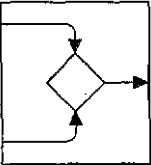
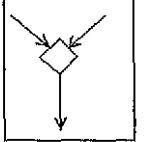
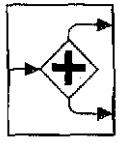
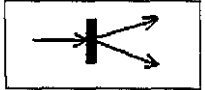
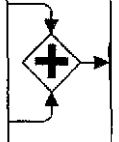

Comparison of Transition/Edges of BPMN and UML Notations					
BPMN 1.0			UML 2.0		
Element	Description	Notation	Element	Description	Notation
Merge	BPMN uses the term "merge" to refer to the exclusive combining of two or more paths into one path (also known as an a OR-Join).		Merge	This node represents a point in an activity where several incoming edges come together into a single outgoing edge.	
Fork	Fork or AND-Split is a place in the Process where activities can be performed concurrently, rather than sequentially.		Fork	This node represents a point in an activity where a single incoming flow is split into several outgoing flows.	
Join	Synchronization or an AND-Join is a place in the Process to combine two or more parallel paths into one path.		Join	This node represents a point in an activity where several incoming flows are synchronized into a single outgoing flow.	

Table B6: Comparison of BPMN and UML Process Patterns

Comparison of Process Patterns using BPMN and UML Notations		
Pattern	BPMN	UML
Sequence		
Parallel Split (And Split)		
Synchronization		
Exclusive Choice (Or Split)		

Table B7: Comparison of BPMN and UML Process Patterns

Comparison of Process Patterns using BPMN and UML Notations		
Pattern	BPMN	UML
Simple Merge		
Multiple Choice		
Multiple Merge		

Comparison of Process Patterns using BPMN and UML Notations		
Pattern	BPMN	UML
Discriminator	<p>Parallel Split (Unclassified Flow)</p> <p>Discriminator Merging Gateway</p>	<p>[B or C completed]</p>
N out of M Join	<p>Parallel Split (Unclassified Flow)</p> <p>N out of M Join Gateway</p>	<p>[Condition]</p>

Table B8: Comparison of BPMN and UML Process Patterns

Table B9: Comparison of BPMN and UML Process Patterns

Comparison of Process Patterns using BPMN and UML Notations		
Pattern	BPMN	UML
Synchronizing Merge		
Arbitrary Cycles		

APPENDIX C

JGRAPH XML SCHEMA

What follows is the XML Schema Definition (XSD) file generated from the JGraph XML output file of the diagram using Altova XMLSpy product.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XMLSpy v2007 rel. 3 spl
(http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xs:element name="void">
    <xs:complexType>
      <xs:choice minOccurs="0">
        <xs:element ref="double"/>
        <xs:element ref="object"/>
        <xs:element ref="void" maxOccurs="unbounded"/>
        <xs:sequence>
          <xs:element ref="string"
maxOccurs="unbounded"/>
          <xs:choice minOccurs="0">
            <xs:element ref="boolean"/>
            <xs:element ref="int"/>
            <xs:element ref="object"/>
          </xs:choice>
        </xs:sequence>
      </xs:choice>
      <xs:attribute name="property" type="xs:string"/>
      <xs:attribute name="method" type="xs:string"/>
      <xs:attribute name="index" type="xs:byte"/>
      <xs:attribute name="id" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="string" type="xs:string"/>
  <xs:element name="object">
    <xs:complexType>
      <xs:choice minOccurs="0">
        <xs:element ref="int" maxOccurs="unbounded"/>
        <xs:sequence>
          <xs:element ref="null" minOccurs="0"/>
          <xs:element ref="void"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:sequence>
          <xs:element ref="string"/>
          <xs:choice>
            <xs:element ref="int"
maxOccurs="unbounded"/>

```



```

maxOccurs="unbounded"/>
    <xs:element ref="void"
    </xs:choice>
</xs:sequence>
<xs:sequence>
    <xs:element ref="object"
maxOccurs="unbounded"/>
    <xs:sequence minOccurs="0">
        <xs:sequence minOccurs="0">
            <xs:element ref="array"
maxOccurs="unbounded"/>
                <xs:element ref="boolean"/>
            </xs:sequence>
        <xs:element ref="void"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:sequence>
    </xs:choice>
    <xs:attribute name="idref" type="xs:string"/>
    <xs:attribute name="id" type="xs:string"/>
    <xs:attribute name="class" type="xs:string"/>
</xs:complexType>
</xs:element>
<xs:element name="null">
    <xs:complexType/>
</xs:element>
<xs:element name="java">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="object"/>
        </xs:sequence>
        <xs:attribute name="version" use="required"
type="xs:string"/>
    <xs:attribute name="class" use="required"
type="xs:string"/>
    </xs:complexType>
</xs:element>
<xs:element name="int" type="xs:short"/>
<xs:element name="double" type="xs:decimal"/>
<xs:element name="boolean" type="xs:boolean"/>
<xs:element name="array">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="void" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="length" use="required"
type="xs:byte"/>
        <xs:attribute name="class" use="required"
type="xs:string"/>
    </xs:complexType>
</xs:element>
</xs:schema>

```

APPENDIX D

WFGENSYSTEM-UNF-VISUAL SYSTEM DEMONSTRATION: UNF HELPDESK EXAMPLE

This appendix presents a demonstration of the WGenSystem-UNF-Visual application, using the VM from the design of the workflow network to the implementation of a complete system. The complete system is assembled by creating a form, creating a workflow, creating a workflow definition to link the form and workflow, and creating a workflow system using the workflow definition. The example here shows how to create a sample workflow system, UNF HelpDesk, using the VM.

As shown in Figure D1, the user logs in to WGenSystem-UNF-Visual application.

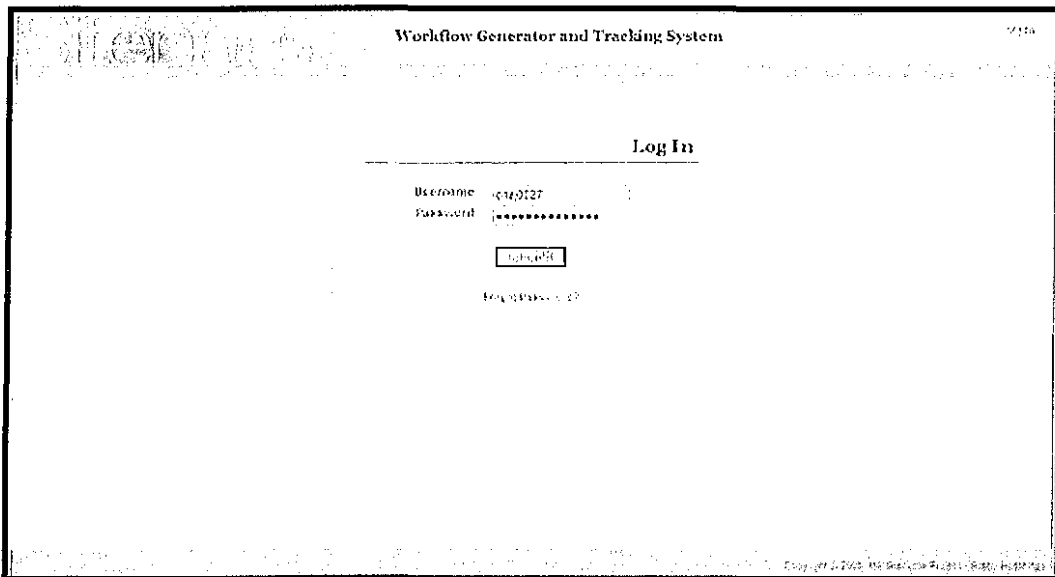


Figure D1: The Login Screen

As shown in Figure D2, the Workflow Engineer can design a form, using the Design Form option from the Form Management menu.

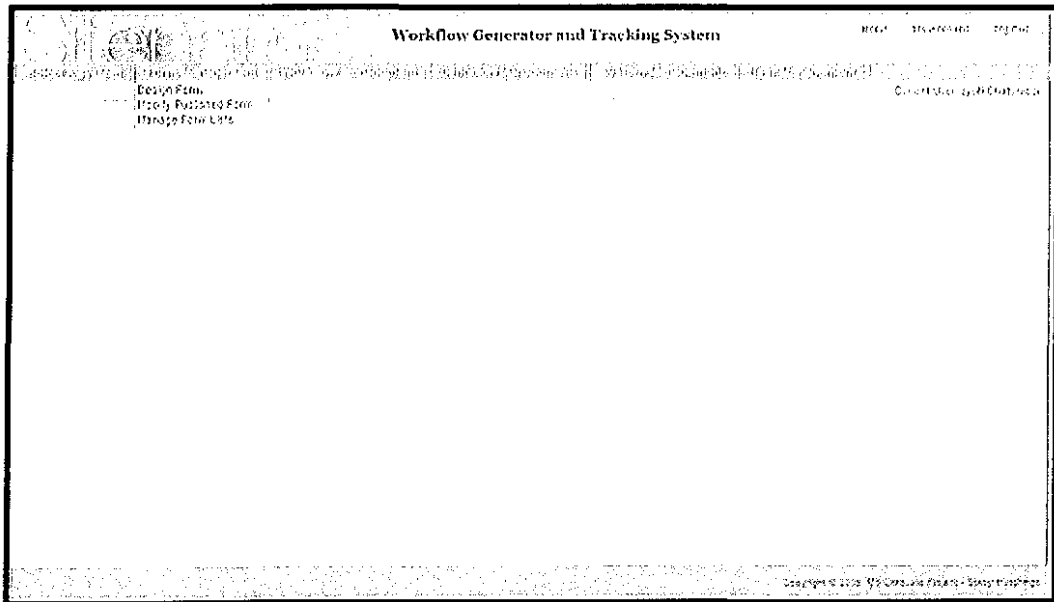


Figure D2: Design Form Option

As shown in Figure D3, the Workflow Engineer enters a form label and optional description, to add a new form. Figure D4 and Figure D5 show that the user adds pages and sections to the form.

Workflow Generator and Tracking System

Design A New Form

Please select a form

Form | Pages | Sections | Controls | Section

Form Label:

Description:

Mode:

Delete Form:

Version:

Created By:

Modified By:

Save Form

Figure D3: Design Form - Form Tab

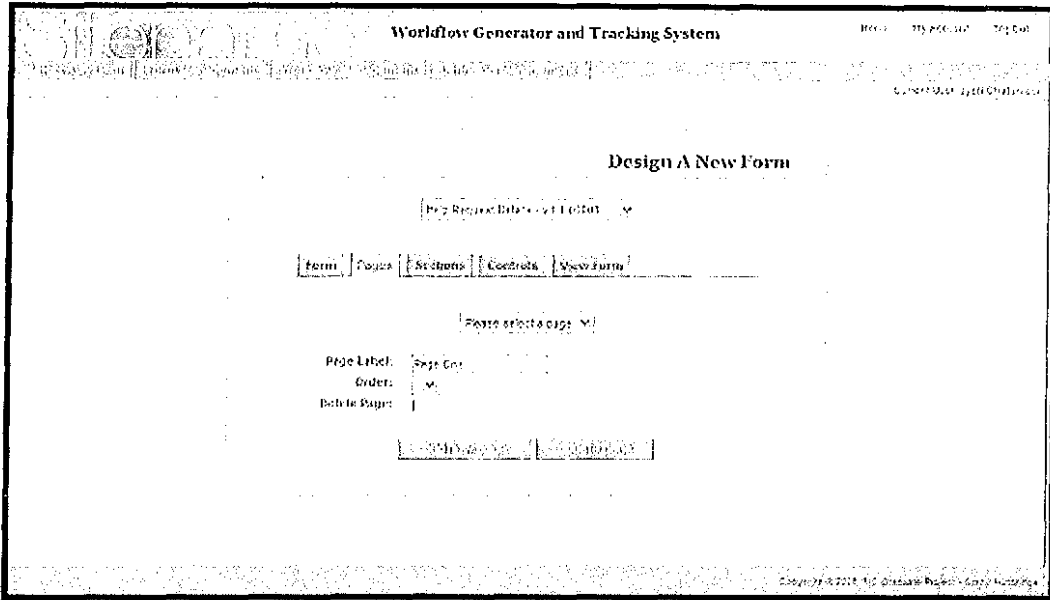


Figure D4: Design Form - Pages Tab

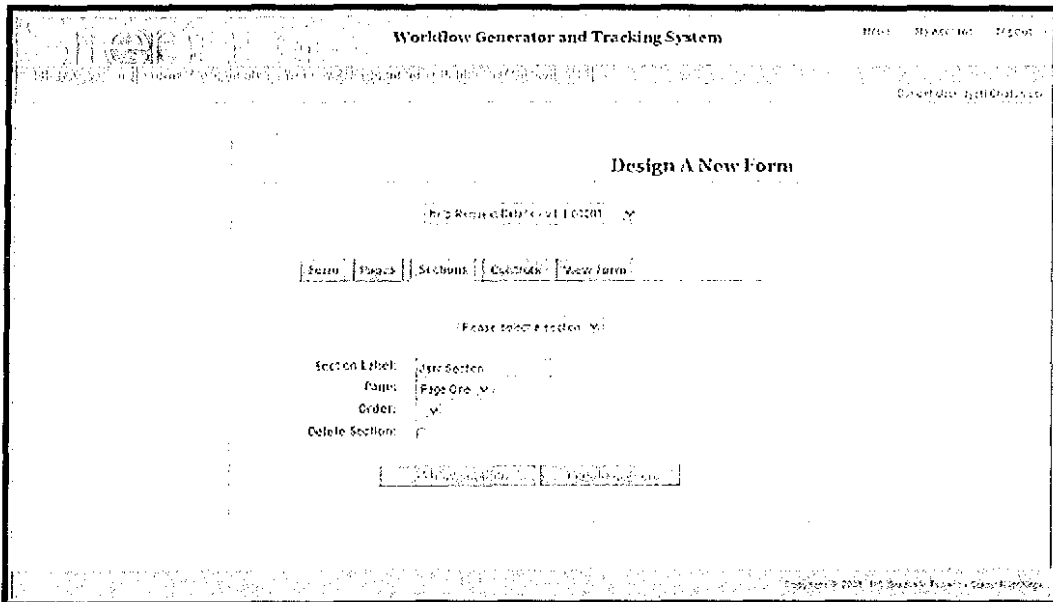


Figure D5: Design Form - Sections Tab

Before adding the controls to the form, the Workflow Engineer creates form list categories to be used on the form, by selecting Manage Form List from the Form Management menu. The Category tab is shown in Figure D6 and the Item tab is shown in Figure D7.

The screenshot displays the 'Workflow Generator and Tracking System' interface. At the top, there is a navigation bar with 'Home', 'My Account', and 'Logout' options. The main heading is 'Create or Modify A Form List'. Below this, there is a dropdown menu for 'Please select a category' and a radio button for 'Active'. The interface is divided into two tabs: 'Category' (selected) and 'Item'. The 'Category' tab contains the following fields:

- Category:** Application Type
- Description:** Type of exceptions to help resolve system
- Active:**
- Created By:** (empty text field)
- Modified By:** (empty text field)

At the bottom of the form, there are two buttons: 'Save' and 'Cancel'. The footer of the page includes a copyright notice: 'Copyright © 2025, All Rights Reserved by [unreadable]'. The page number '- 67 -' is located at the bottom center of the document.

Figure D6: Manage Form Lists - Category Tab

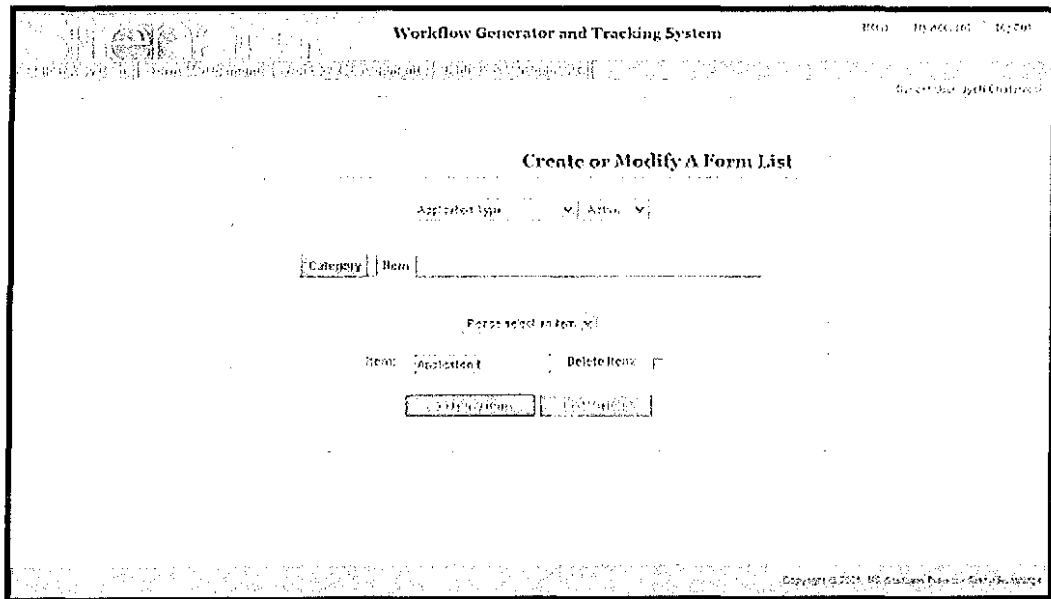


Figure D7: Manage Form Lists - Item Tab

As shown in Figure D8, the Workflow Engineer adds controls to the form. The form can be viewed, as shown in Figure D9.

The screenshot shows the 'Design A New Form' application with the 'Controls' tab selected. At the top, there is a title bar 'Design A New Form' and a menu bar with 'File', 'Request Details - v.1.0.02101', 'Form', 'Pages', 'Sections', 'Controls', and 'View Form'. Below the menu bar is a 'Please select a control' dropdown menu. The main area contains the following configuration options:

- Control Type: TEXT
- Select List: None
- Control Label: User Name
- Tooltip: Name of the User
- Default Value: (empty)
- Width: 15
- Height: (empty)
- Section: Page One - User Section
- Order: 1
- Delete Control:

At the bottom, there are two buttons: 'OK' and 'Cancel'.

Figure D8: Design Form - Controls Tab

The screenshot shows the 'Design A New Form' application with the 'View Form' tab selected. The form is displayed within a window titled 'Request Details - v.1.0.02101'. The form contains the following fields:

- User Name: Application A
- Application Details: (empty)
- Tech Support Section: Support Dept

At the bottom, there is a 'View Form' button.

Figure D9: Design Form - View Form Tab

The Workflow Engineer can define roles, to be used in the workflow, from the Manage Roles option of the Workflow Management menu, as shown in Figure D10.

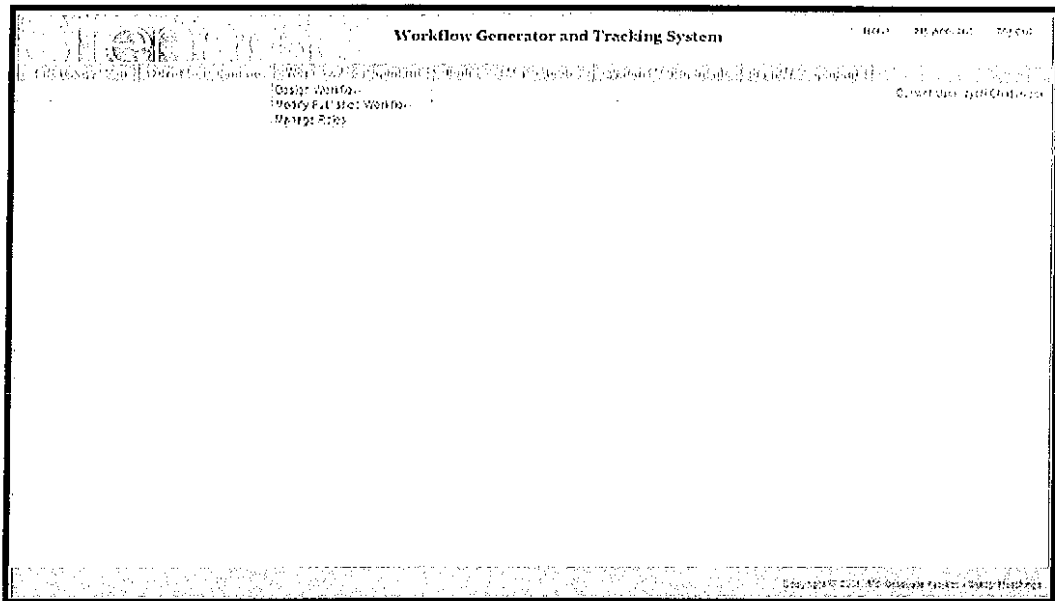


Figure D10: Manage Roles Option

As shown in Figure D11, the Workflow Engineer creates a custom role and, in Figure D12, adds Account Members to that role. Group members can also be added to the role, as shown in Figure D13.

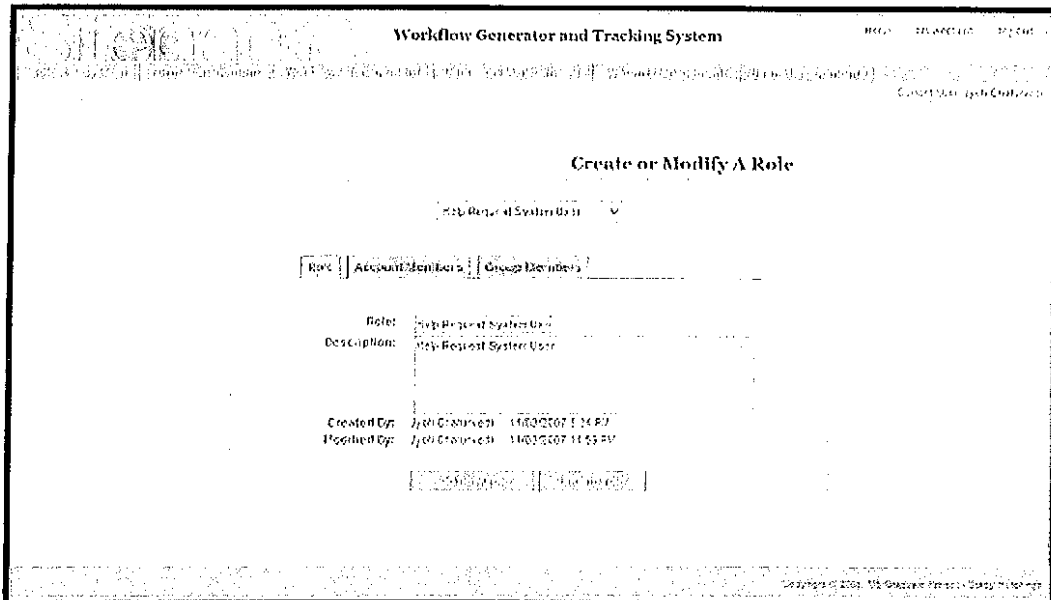


Figure D11: Manage Roles -Roles Tab

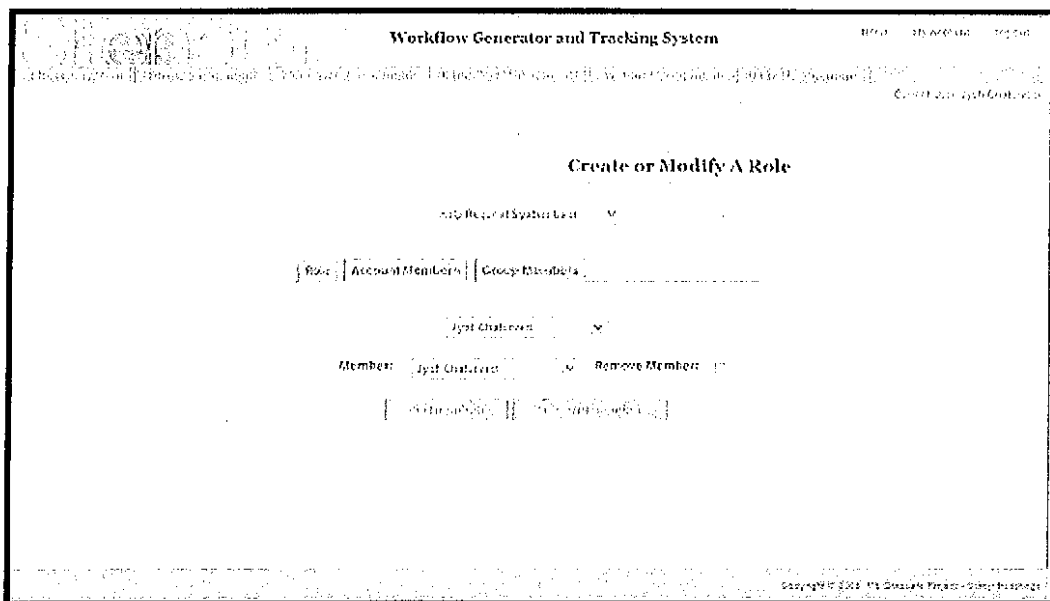


Figure D12: Manage Roles - Account Members Tab

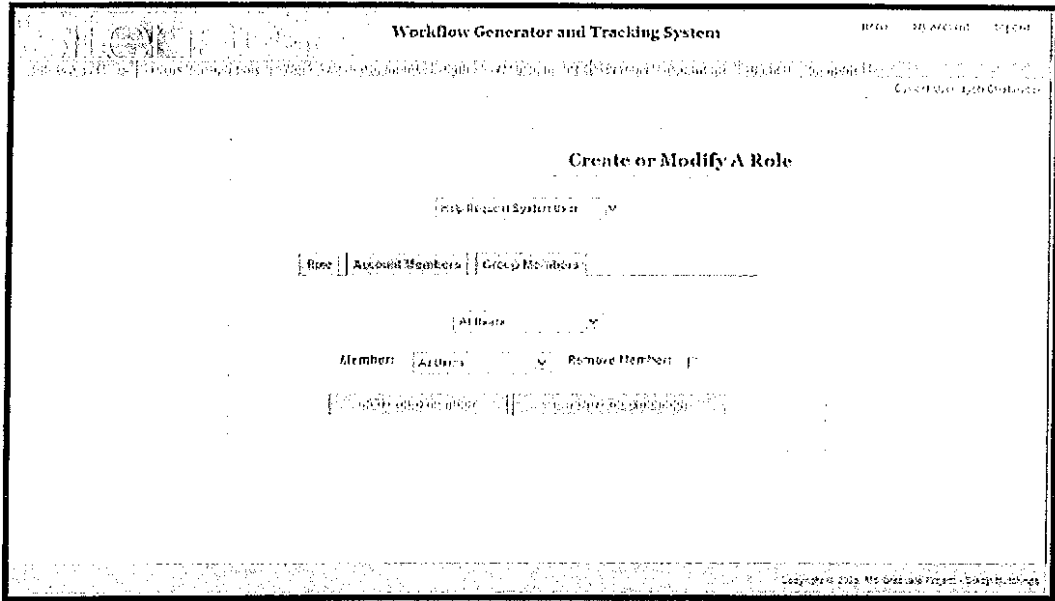


Figure D13: Manage Roles - Group Members Tab

Then the Workflow Engineer designs the workflow, using the Design Workflow option from the Workflow Management menu (Figure D14).

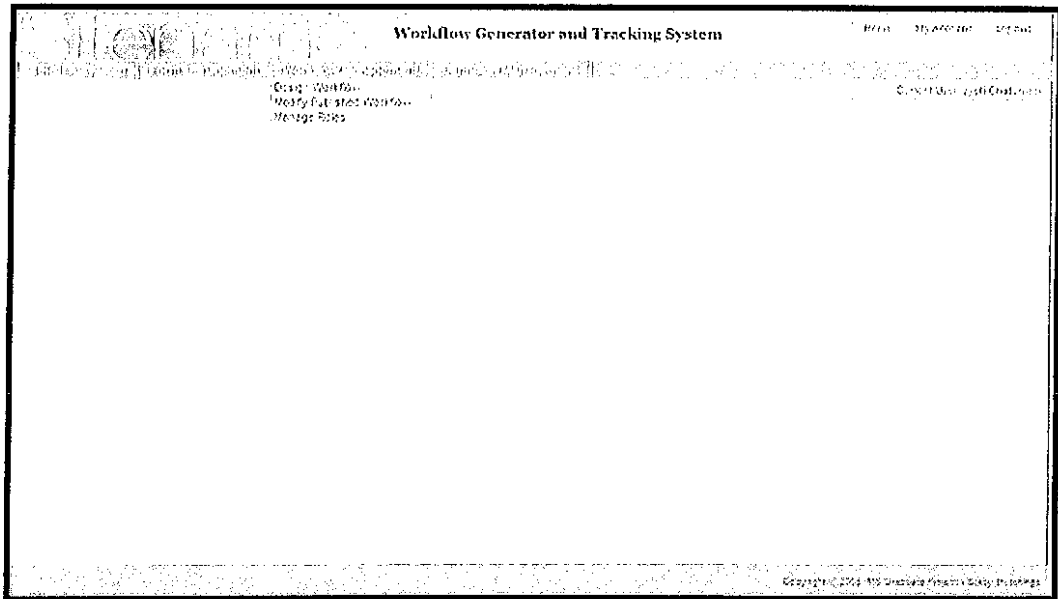


Figure D14: Design Workflow Option

The Workflow Engineer can modify an existing workflow, by selecting it from the dropdown list. She can create a new workflow, by selecting the option New Workflow, as shown in Figure D15.

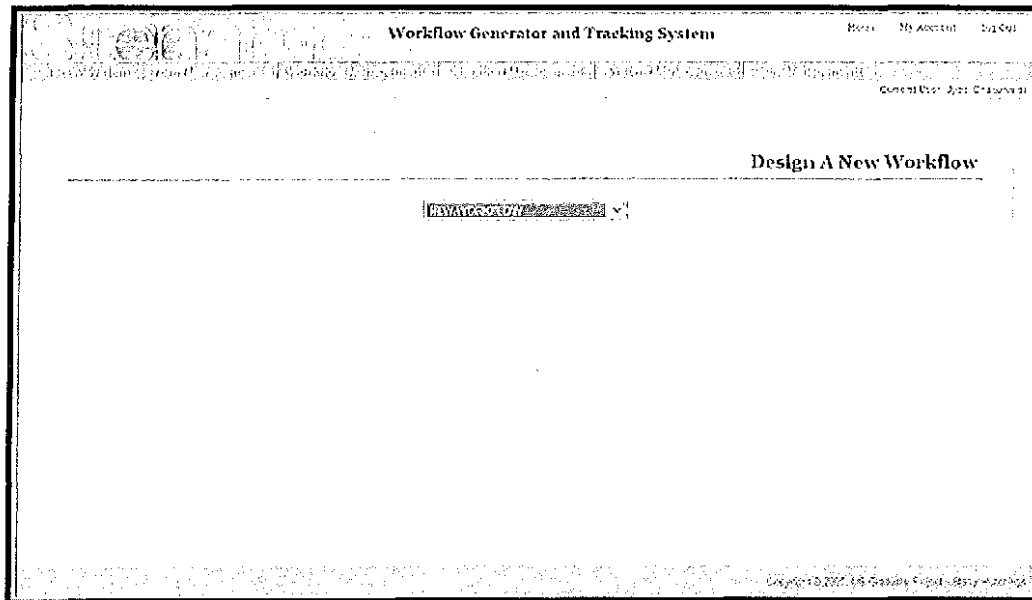


Figure D15: Create New Workflow

The VM starts and the Workflow Properties dialog is displayed. The Workflow Engineer enters the workflow name and description for the new workflow, as shown in Figure D16.

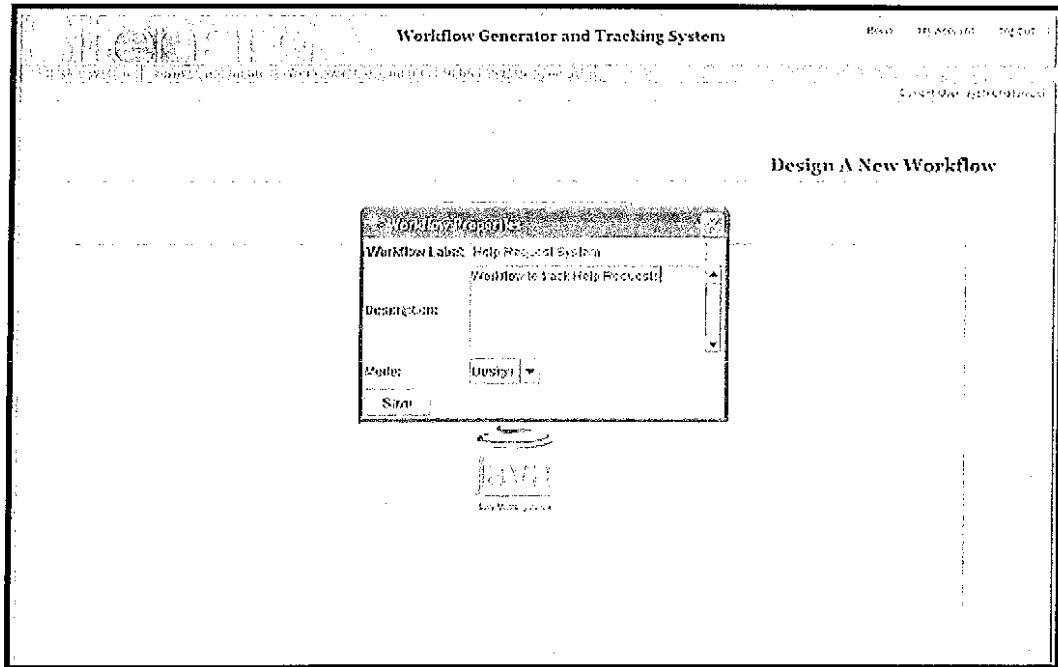


Figure D16: Workflow Properties Dialog

The Workflow Engineer can design the workflow, using the VM toolbar as shown in Figure D17.

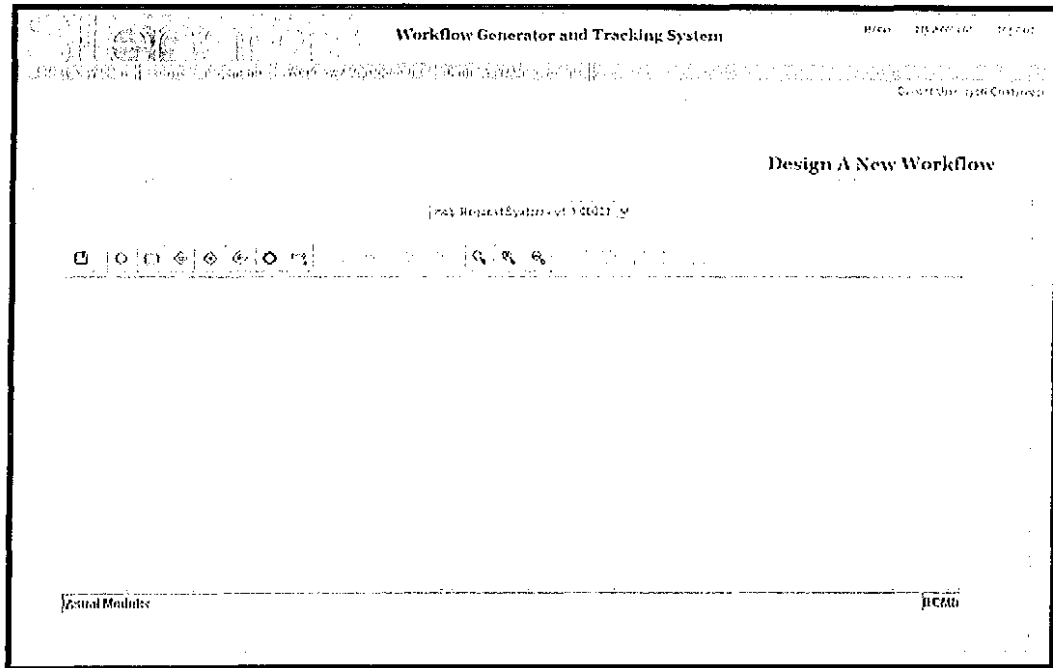


Figure D17: VM

The Workflow Engineer designs the workflow, by laying out the processes (states) and flows (transitions), as shown in Figure D18.

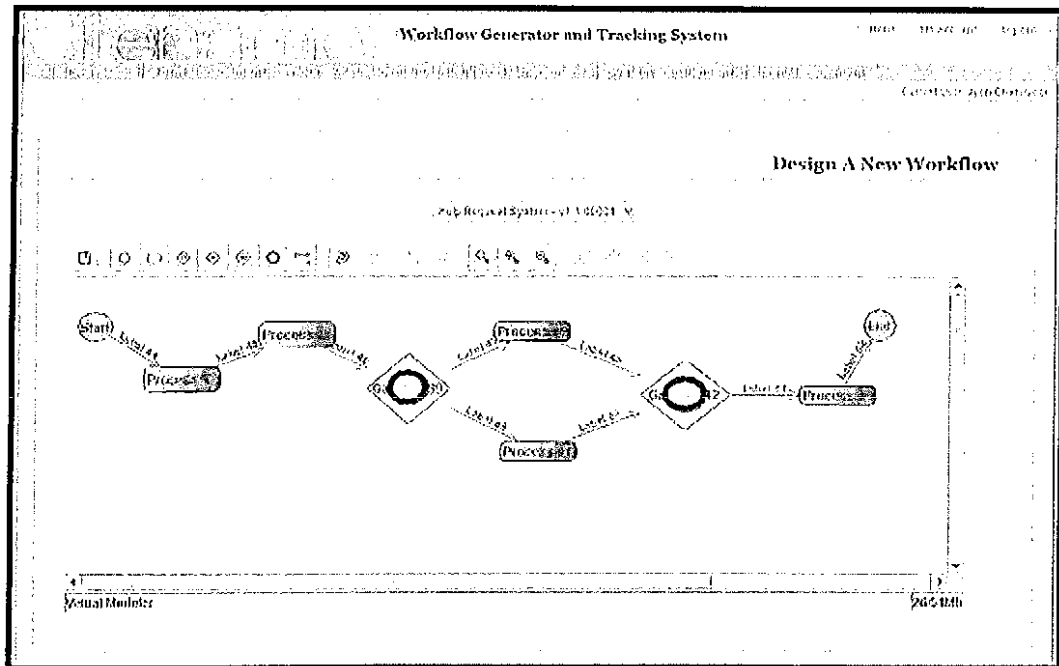


Figure D18: Initial Workflow Design

The Workflow Engineer saves the workflow design and starts defining the process properties, as shown in Figure D19. The Workflow Engineer then defines the flow properties, as shown in Figure D20.

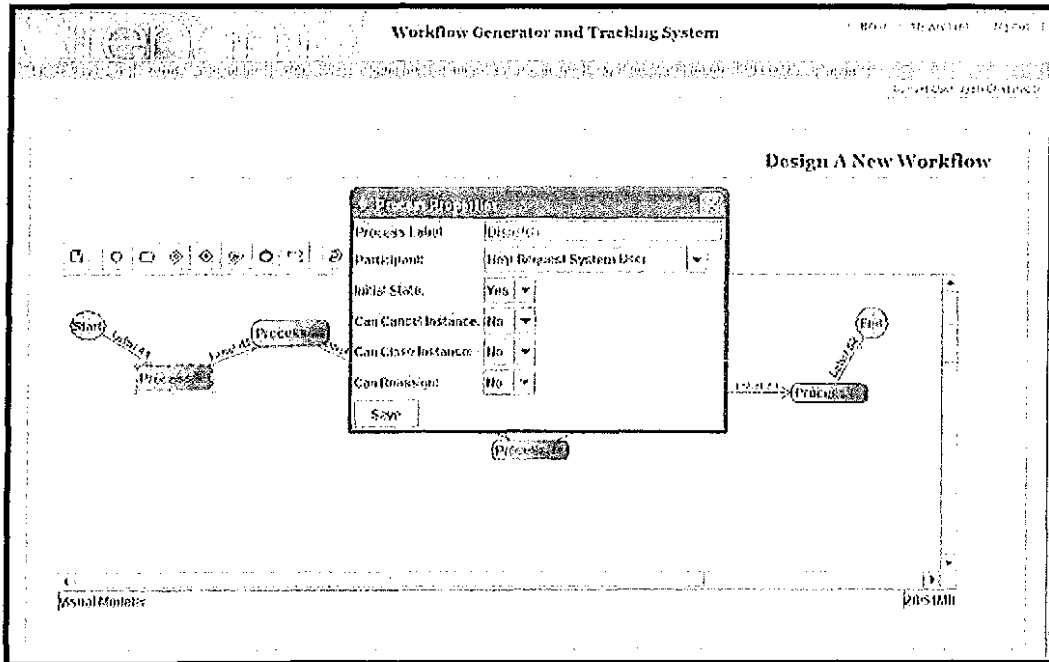


Figure D19: Process Properties Dialog

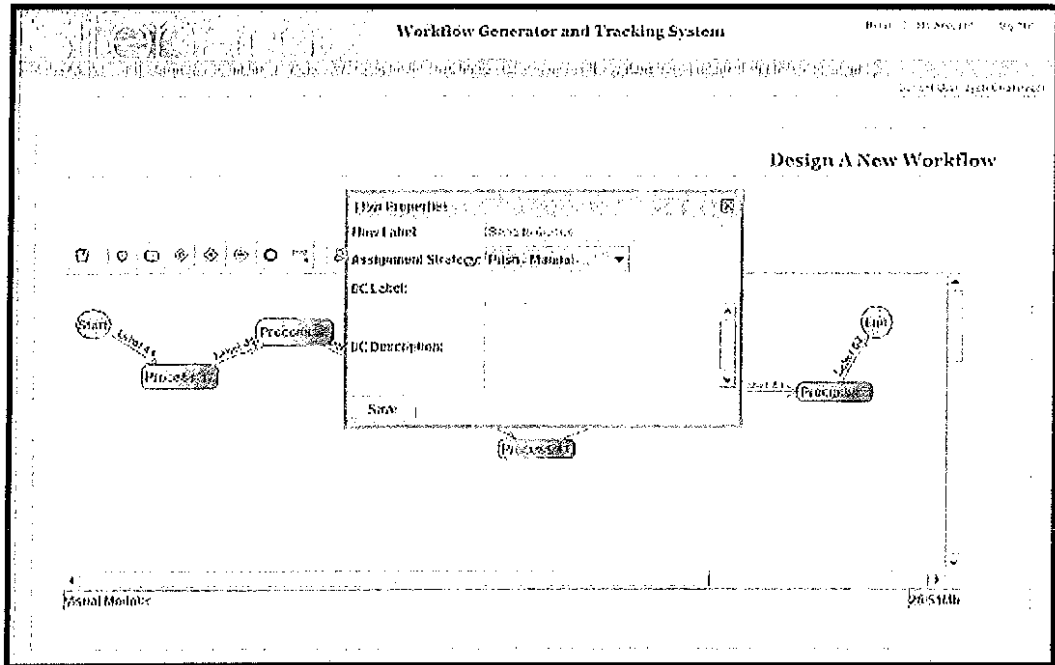


Figure D20: Flow Properties Dialog

After the workflow process and flow properties are saved, the workflow design can be moved to Test mode, as shown in Figure D21, in order to create a workflow definition.

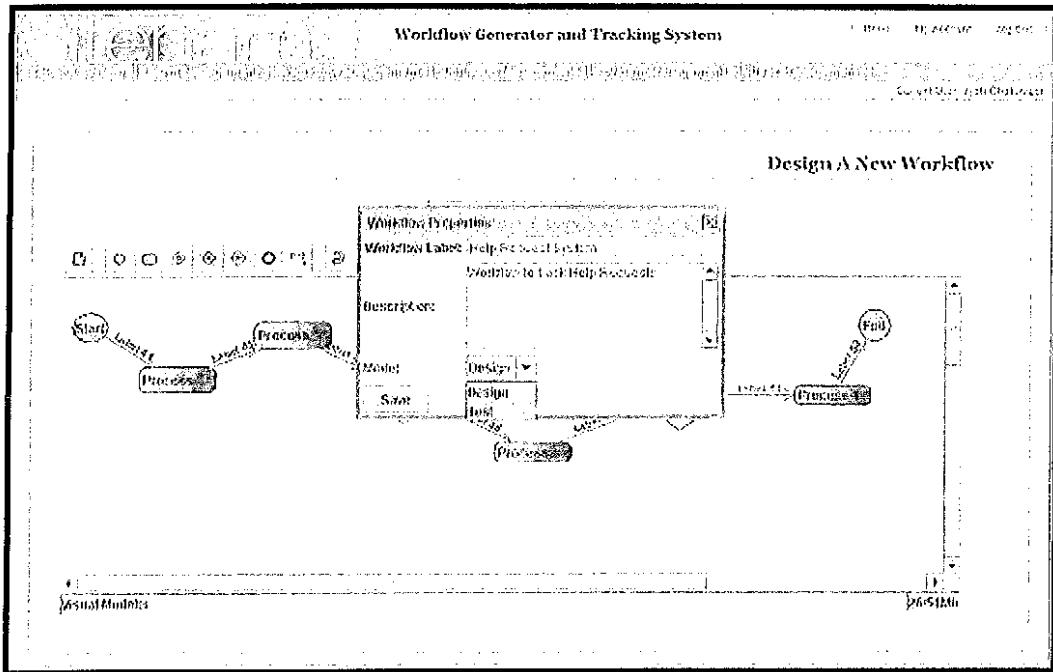


Figure D21: Workflow Mode Changed To Test

The final designed workflow diagram for the HelpDesk System is shown in Figure D22.

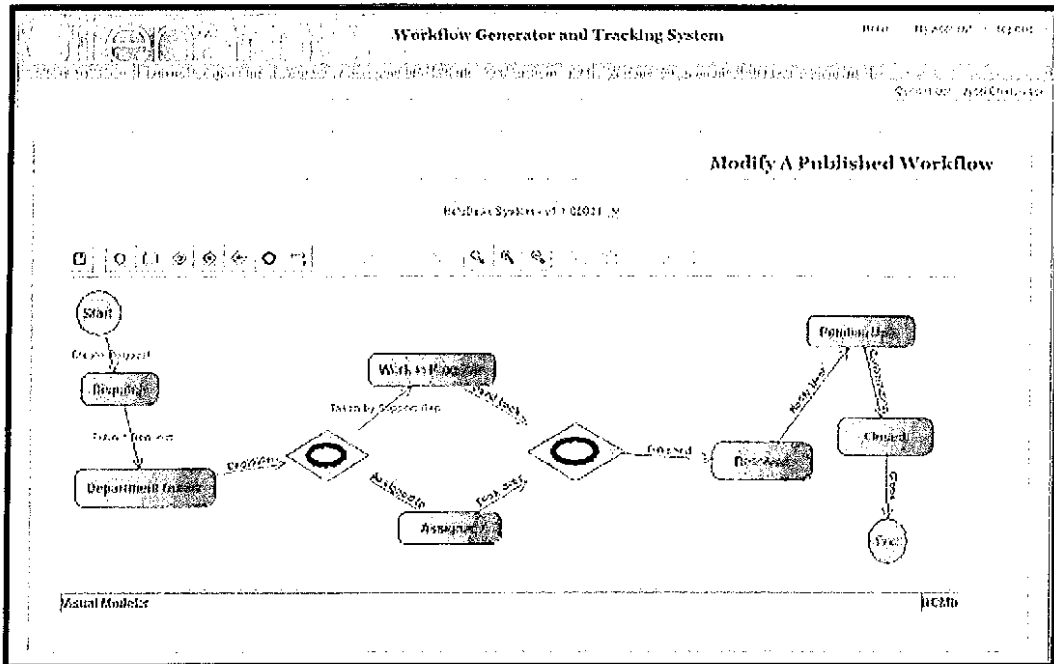


Figure D22: HelpDesk System Workflow Diagram

The Workflow Engineer then creates a process definition to associate the design Help Request Details form and the HelpDesk workflow, by selecting Design Definition from the Definition Management menu, as shown in Figure D23.

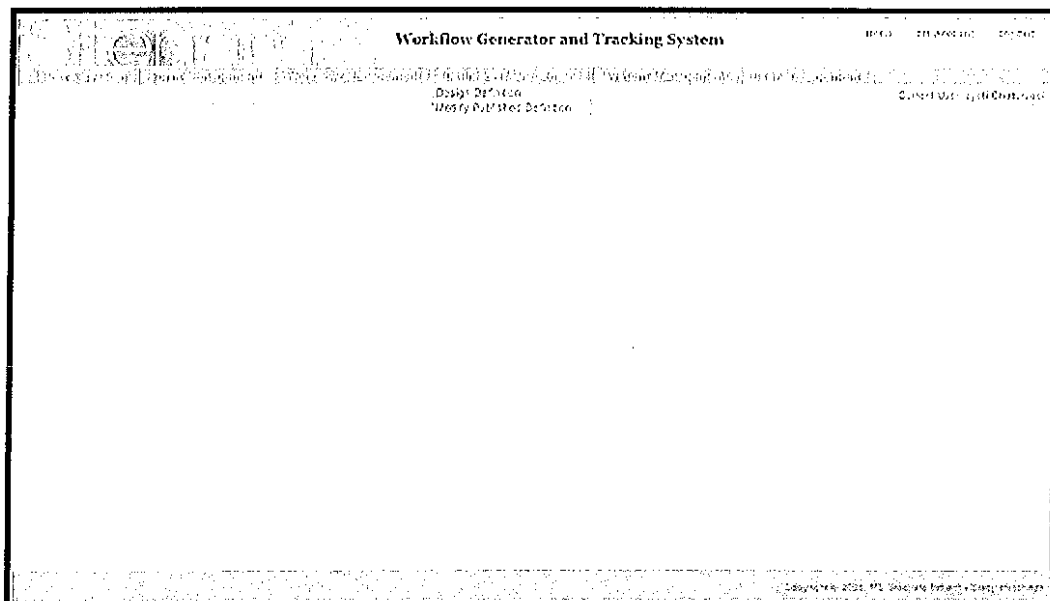


Figure D23: Definition Management - Design Definition

The Workflow Engineer creates a definition label and description, and then selects the Help Request Details form and the HelpDesk System workflow, as shown in Figure D24.

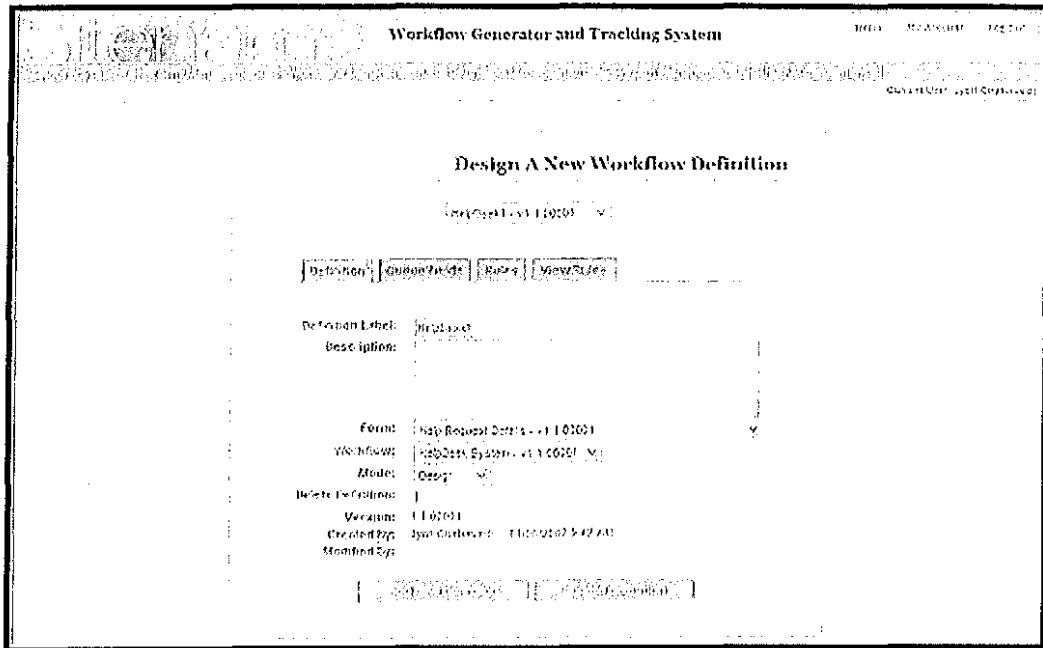


Figure D24: Design Definition – Definition Tab

The Workflow Engineer selects the fields to display on the pending tasks page from the Queue Fields tab, as shown in Figure D25.

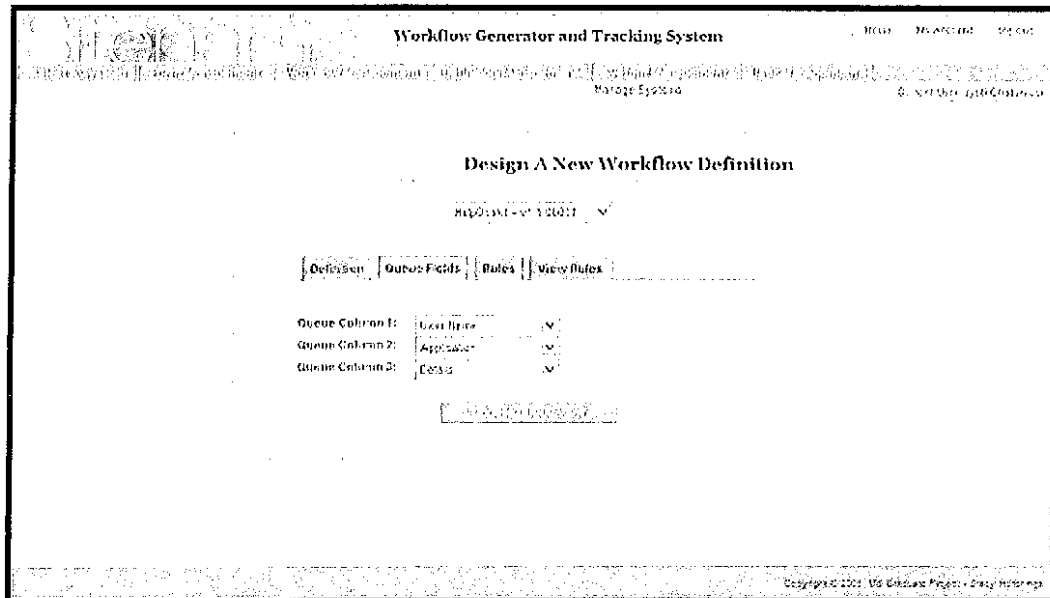


Figure D25: Design Definition - Queue Fields Tab

The Workflow Engineer creates the form rules and transition rules from the Rules tab, as shown in Figure D26. All the rules can be viewed from the View Rules tab, as shown in Figure D27.

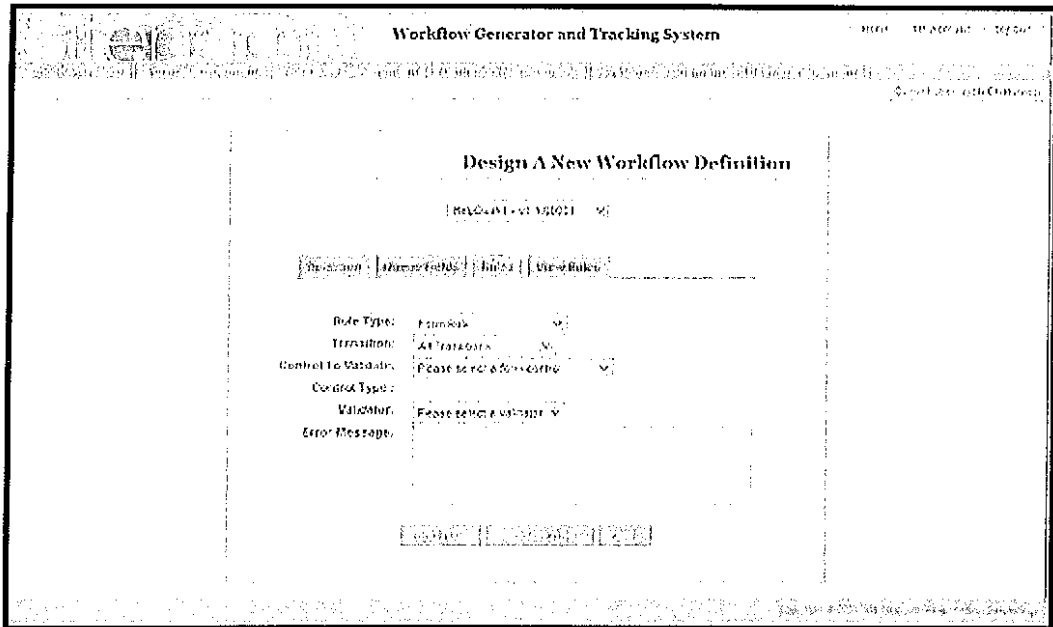


Figure D26: Design Definition - Rules Tab

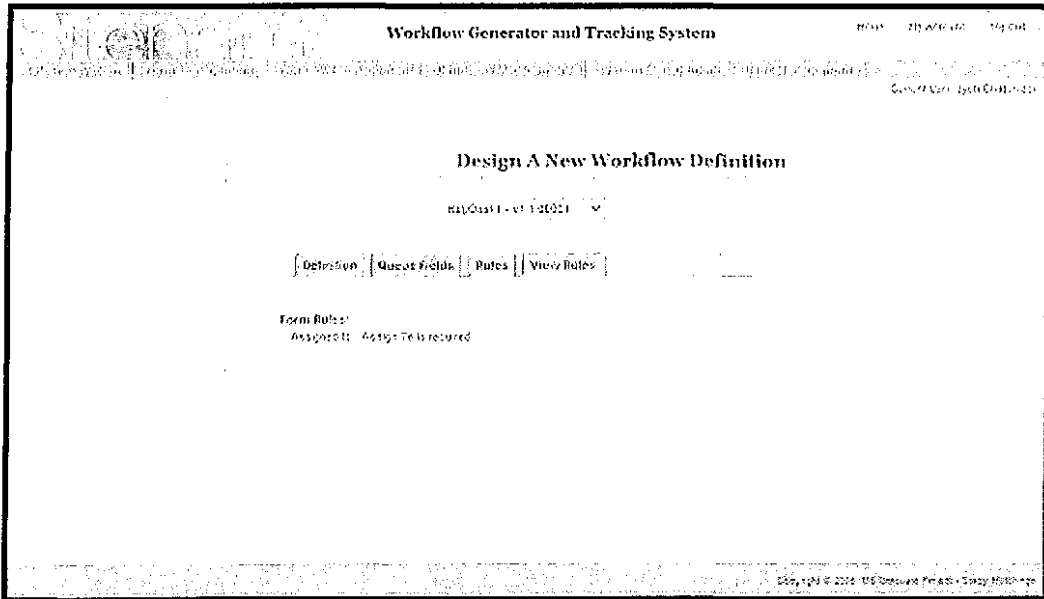


Figure D27: Design Definition - View Rules Tab

The Workflow Engineer publishes the process definition. The Administrator then creates the system definition, by selecting Manage Systems from the System Management menu, as shown in Figure D28.

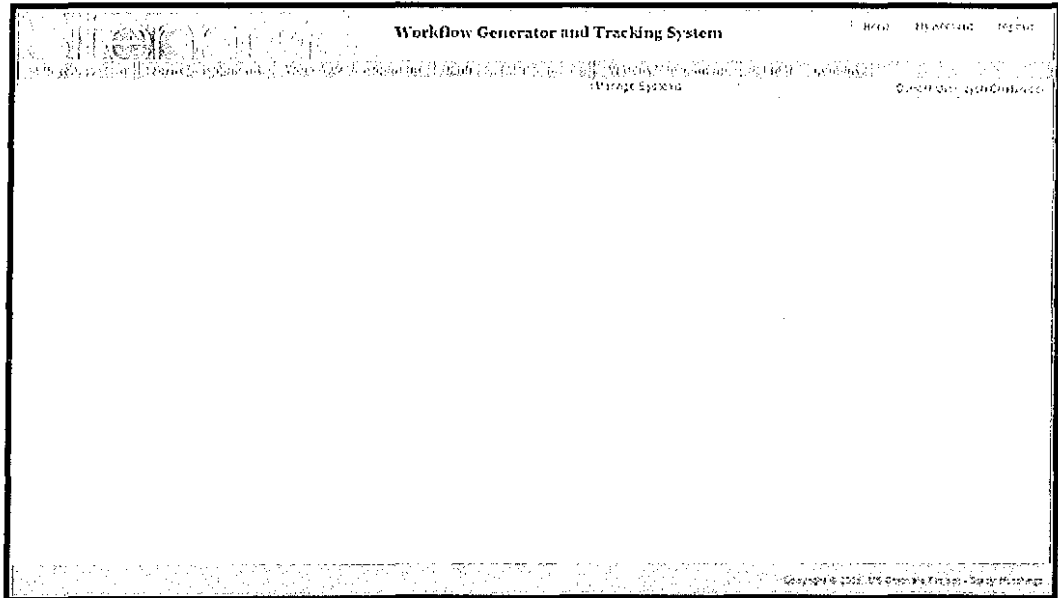


Figure D28: System Management – Manage System

The Administrator defines the system label, description, and style sheet to be used with the system, on the System tab, as shown in Figure D29.

The screenshot shows a web application window titled "Workflow Generator and Tracking System". The main content area is titled "Create or Modify A System". At the top, there is a "Name:" field with the value "System". Below this, there are several input fields:

- System Label:** [Text Field]
- Description:** [Text Field]
- Page Title:** [Text Field]
- Stylesheet:** [Text Field]
- Access:** [Text Field]

At the bottom of the form, there are two lines of text indicating creation and modification details:

Created By: [User Name] [Timestamp]
Modified By: [User Name] [Timestamp]

At the bottom right of the window, there is a footer: "Copyright © 2006, All Rights Reserved. System Generator".

Figure D29: Manage System - System Tab

The Administrator associates the published process definition with the system, as shown in Figure D30.

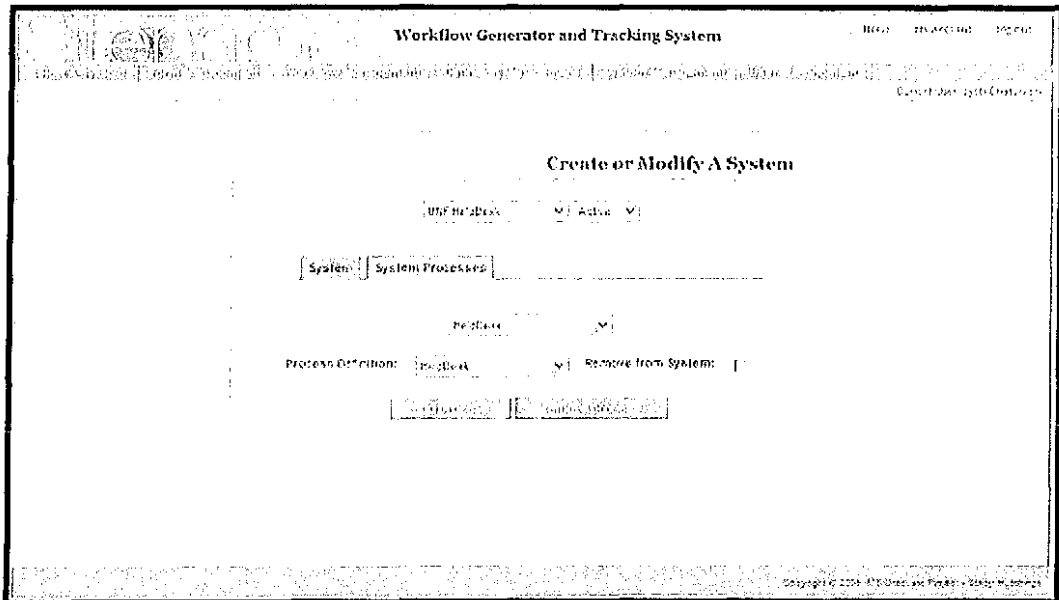


Figure D30: Manage System - System Processes Tab

The created UNF HelpDesk System is then available in the Select a System menu, as shown in Figure D31.

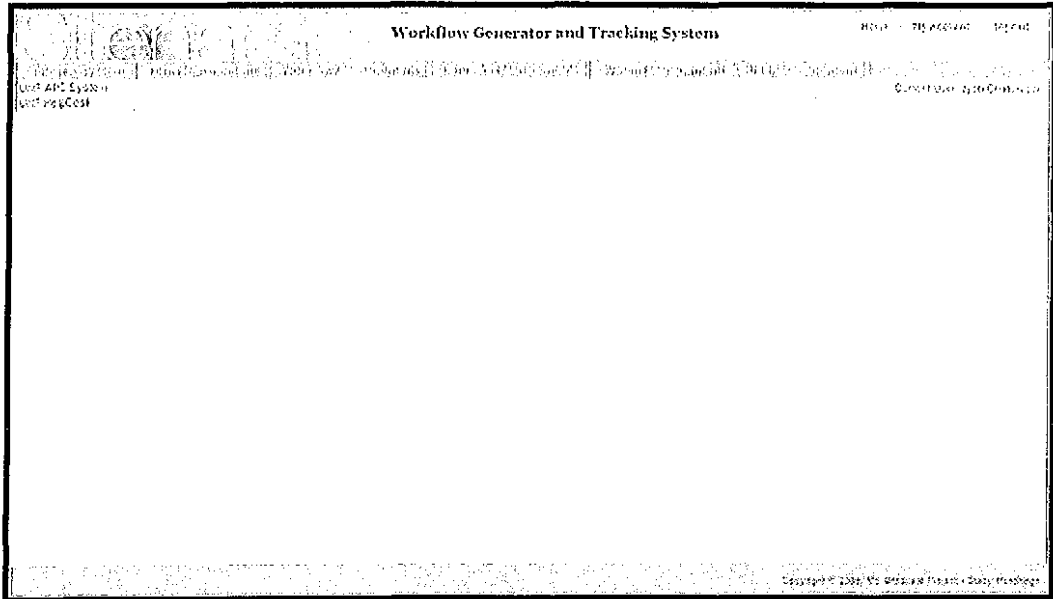


Figure D31: Select a System - UNF HelpDesk

The User selects a system to get the task management component, as shown in Figure D32.

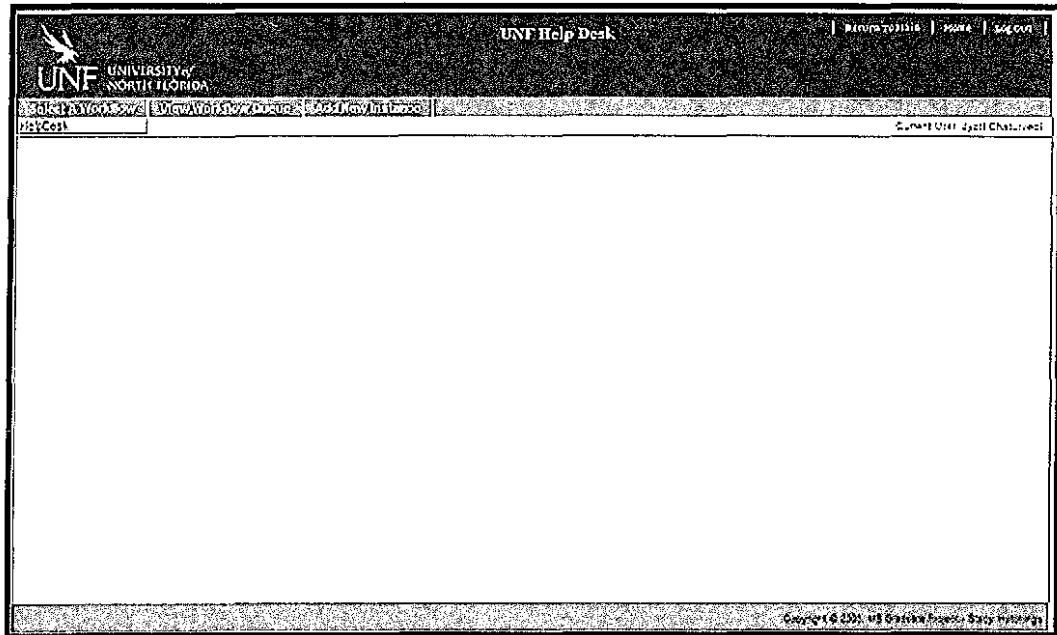


Figure D32: UNF HelpDesk - Select a Workflow

The User adds a new workflow instance, by selecting the Add New Instance menu. The system presents the User with the Help Request Details form, as shown in Figure D33.

The screenshot shows a web browser window with the title 'UNF Help Desk'. The browser's address bar contains 'http://www.unf.edu/helpdesk/'. The page header includes the UNF logo and 'UNIVERSITY of NORTH FLORIDA'. The main content area is titled 'HelpDesk - Help Request Details'. It features a 'User Section' with a 'User Name' field, an 'Application' dropdown menu set to 'App12091A', and a 'Details' field. Below this is a 'Team Support Section' with an 'Assign To' dropdown menu set to 'SupportRes1'. A 'Add New Instance' button is located at the bottom of the form. The footer of the page contains the text 'Copyright © 2001. All rights reserved. Helpdesk - Support Request'.

Figure D33: UNF HelpDesk - Add New Instance

User Task Management lists the new instance (ID 14) with initial status of Dispatch, as shown in Figure D34.

The screenshot shows the UNF HelpDesk interface. At the top, there is a header with the UNF logo and 'UNIVERSITY OF NORTH FLORIDA'. Below the header, there are navigation tabs: 'Select Workflow', 'View Workflow Codes', and 'Add New Instance'. The main content area displays 'HelpDesk status: Pending Review' with a dropdown arrow. Below this is a table with the following data:

ID	Submit Date	Submit By	Version	Status	Assigned To
13	11/04/2007 12:47 AM	Jyoti Chaturvedi	1.1.00001	Resolved	Jyoti Chaturvedi
14	11/04/2007 10:41 PM	Jyoti Chaturvedi	1.1.00001	Dispatch	Jyoti Chaturvedi

Figure D34: UNF HelpDesk - Pending Queue

The User selects the ID of the instance, to view the task, as shown in Figure D35. The Summary tab displays the details of the form. The Comments tab displays the history of the instance, as shown in Figure D36.

The screenshot shows the UNF Help Desk interface. At the top, there is a navigation bar with the UNF logo and the text 'UNF UNIVERSITY OF NORTH FLORIDA'. The main header area contains 'UNF Help Desk' and user information: 'Admin 7/11/12 | Home | Logout'. Below this is a breadcrumb trail: 'Home > My Account > My Open Cases > My Open Cases'. The main content area has a tabbed interface with 'Summary' selected, and other tabs for 'Comments', 'Transition', and 'Resubmit'. The instance details are as follows:

- ID: 18
- Submitted By: Jyoti Chakraborty
- Submitted On: 11/04/2012 07:16:41 PM
- Current Status: In Progress
- Assigned To: Jyoti Chakraborty

The title of the instance is 'HelpDesk - Help Request Details'. The 'User Section' is defined as:

- Name: Jyoti Chakraborty
- Application: Applications A
- Details: Installation problem with Applications A

The 'Team Support Section' is assigned to 'Support Dept A'. At the bottom of the instance details, there is a 'Resubmit Instance' button.

Figure D35: View Instance Details - Summary Tab

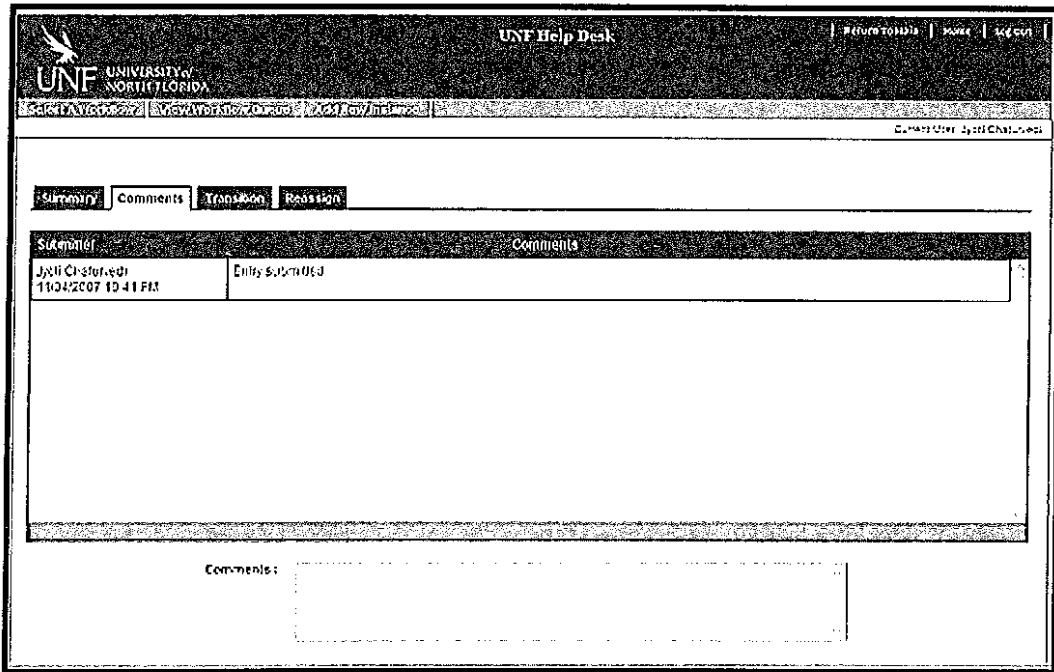


Figure D36: View Instance Details - Comments Tab

The Transition Tab shows transition choices to the User, as shown in Figure D37. The User sends the request to the Department Queue and assigns it to Support Repl, as shown in Figure D38.

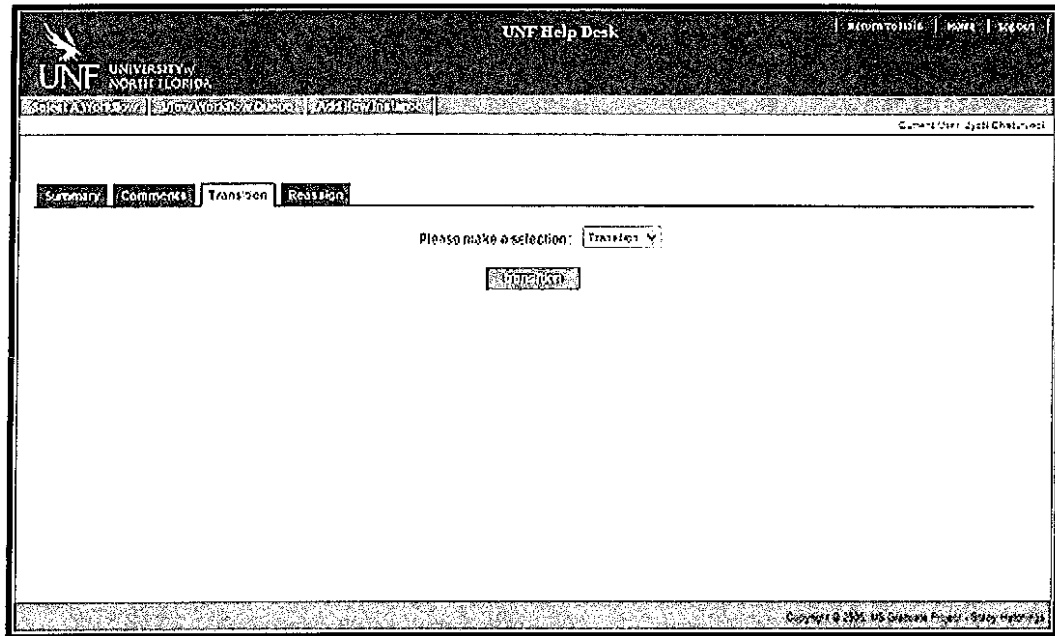


Figure D37: View Instance Details - Transition Tab

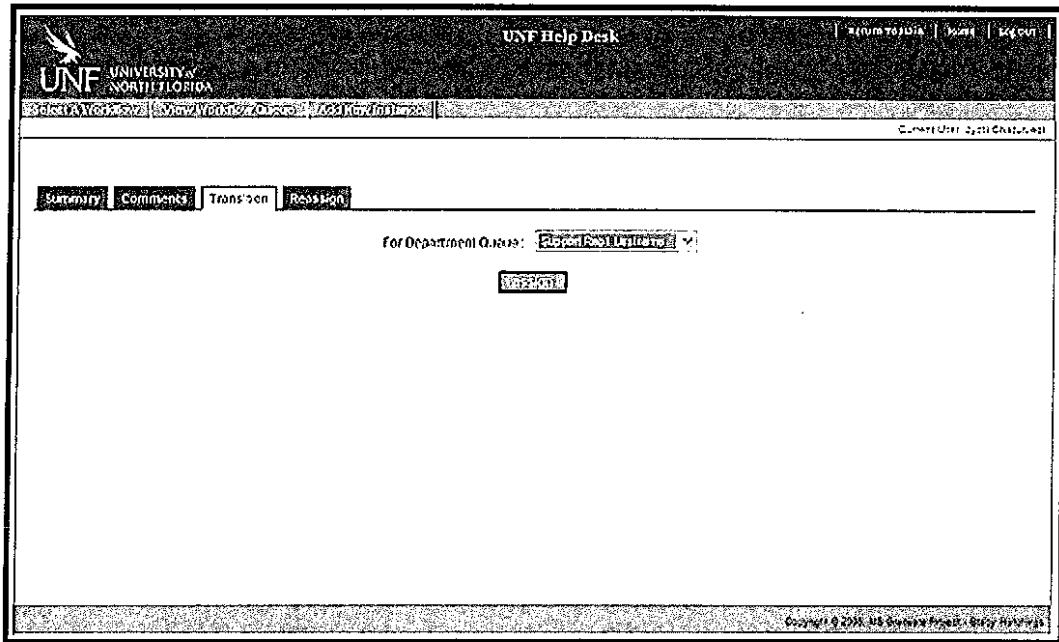


Figure D38: View Instance Details - Transition Tab

Selected instance is then removed from the pending tasks of the User, as shown in Figure D39. Support Repl logs in and views his pending tasks. The workflow instance ID 14 is listed, as shown in Figure D40.

UNF UNIVERSITY OF NORTH FLORIDA

UNF Help Desk

HelpDesk status: [Pending My Review v]

ID	Submit Date	Submit By	Version	Status	Assigned To
13	11/14/2007 12:47 AM	Jyoti Chakraborty	1.0.0.0.0.1	Resolved	Jyoti Chakraborty

Copyright © 2007, US @ Allstate, Phoenix - 844-2-HELP-1234

Figure D39: Pending Tasks for User

UNF Help Desk

UNF UNIVERSITY OF NORTH FLORIDA

HelpDesk status: Pending Review

ID	Submit Date	Submit By	Version	Status	Assigned To
14	11/14/2007 10:41 PM	Josh Chiswick	1.1.00001		Department Code Support Rep Lastname

Copyright © 2005, All Services Programs, Inc. All Rights Reserved

Figure D40: Pending Tasks for Support Rep1

Support Repl transitions the instance to Work In Progress, while working on the request. The pending task is listed in the Work In Progress queue, as shown in Figure D41.

The screenshot shows the UNF Help Desk interface. At the top, there is a header with the UNF logo and the text 'UNIVERSITY OF NORTH FLORIDA'. The main content area displays the 'HelpDesk status: Pending Review (N)'. Below this, there is a table with the following data:

ID	Submit Date	Submit By	Version	Status	Assigned To
12	11/04/2007 10:43 PM	Jay Chismodi	1.100001	Work in Progress	SupportRepl Lashome

At the bottom right of the interface, there is a small copyright notice: 'Copyright © 2007, All Rights Reserved by HelpDesk'.

Figure D41: Pending Tasks for Support Repl

Support Repl transitions the instance to Resolved, when the task is complete. The request is listed as Resolved in the pending tasks queue for the User, as shown in Figure D42.

UNF UNIVERSITY OF NORTH FLORIDA

UNF Help Desk

HelpDesk status:

ID	Submit Date	Submit By	Version	Status	Assigned To
13	11/04/2007 12:47 AM	Josh Chisumedi	1.1.00001	Resolved	Josh Chisumedi
14	11/04/2007 10:41 PM	Josh Chisumedi	1.1.00001	Resolved	Josh Chisumedi

Figure D42: Pending Tasks for User

The User updates the request to Pending User status, while verifying the resolution. The request is listed as Pending User, in the pending tasks queue for the User, as shown in Figure D43.

The screenshot shows the UNF Help Desk interface. At the top, there is a navigation bar with 'UNF UNIVERSITY OF NORTH FLORIDA' on the left and 'UNF Help Desk' in the center. To the right of the navigation bar are links for 'Return to Main', 'Home', and 'Logout'. Below the navigation bar is a breadcrumb trail: 'Request > My Tasks > Pending Tasks'. On the right side of the page, the user's name 'Current User: Jyoti Chakraborty' is displayed. The main content area shows 'HelpDesk status: Pending by Review'. Below this is a table with the following data:

ID	Submit Date	Submitted By	Version	Status	Assigned To
13	11/04/2007 12:47 AM	Jyoti Chakraborty	1.1.00001	Resolved	Jyoti Chakraborty
14	11/04/2007 10:41 PM	Jyoti Chakraborty	1.1.00001	Pending User	Jyoti Chakraborty

At the bottom right of the page, there is a copyright notice: 'Copyright © 2007, US Online People, Inc. All rights reserved.'

Figure D43: Pending Tasks for User

The User updates the request to Closed status, when the request is verified. The request is listed as Closed in the pending tasks queue for the User, as shown in Figure D44.

UNF UNIVERSITY OF NORTH FLORIDA

UNF Help Desk

HelpDesk status: [Pending My Review ▼]

ID	Submit Date	Submitted By	Version	Status	Assigned To
13	11/04/2007 12:47 AM	Jodi Chastaned	1.1.00001	Resolved	Jodi Chastaned
14	11/04/2007 10:41 PM	Jodi Chastaned	1.1.00001	Closed	Jodi Chastaned

Copyright © 2005, MS Global, Inc. All Rights Reserved

Figure D44: Pending Tasks for User

The User can view the complete status log of the request from the Comments tab, as shown in Figure D45.

The screenshot shows the UNF Help Desk interface. At the top, there is a header with the UNF logo and the text 'UNF UNIVERSITY OF NORTH FLORIDA'. Below the header, there are navigation tabs: 'Summary', 'Comments', 'Transition', and 'Resign'. The 'Comments' tab is currently selected. Below the tabs is a table with two columns: 'Submitter' and 'Comments'. The table contains six rows of data, each representing a task update. Below the table, there is a 'Comments:' label followed by a text input field.

Submitter	Comments
Jill Charnock 11/05/2007 12:29 AM	Entry transferred from Pending User to Closed
Jill Charnock 11/05/2007 12:28 AM	Entry transferred from Received to Pending User
Subject Rep: LASHAM 11/05/2007 12:27 AM	Entry transferred from Work In Progress to Resolved
Subject Rep: LASHAM 11/05/2007 12:10 AM	Entry transferred from Department Queue to Work In Progress
Jill Charnock 11/04/2007 11:03 PM	Entry transferred from Dispatch to Department Queue
Jill Charnock 11/04/2007 10:41 PM	Entry submitted

Comments:

Figure D45: Task History

APPENDIX E

XFLOW SYSTEM DEMONSTRATION: SIMPLEWORKFLOW EXAMPLE

What follows is a demonstration of the XFlow application, using the VM to design a workflow. The complete process consists of creating a workflow, deploying the workflow on the JBoss Server, and executing the workflow. The example shows how to create a simple workflow, using the VM.

The User starts the VM applet, as shown in Figure E1.

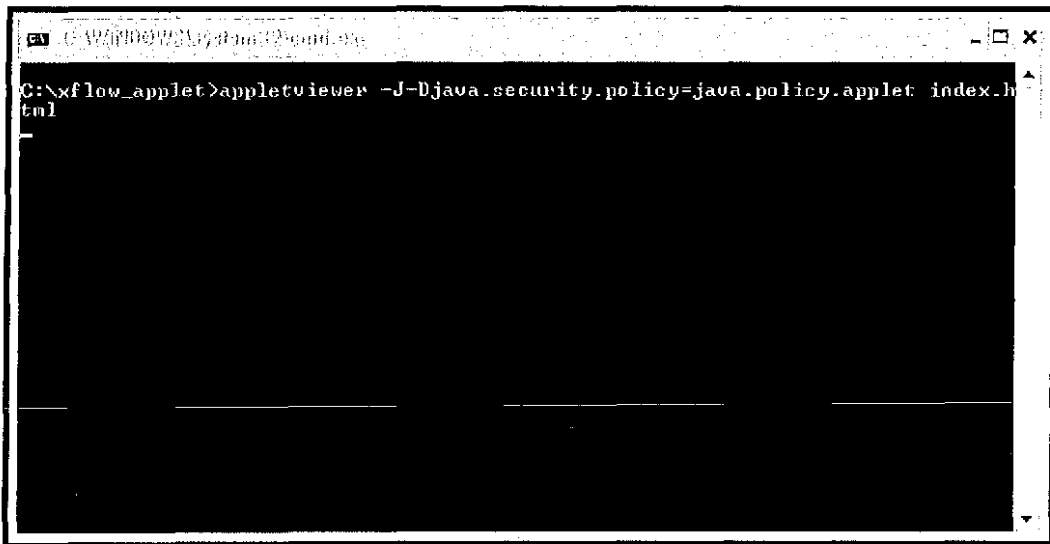


Figure E1: Start VM using Appletviewer

The User designs a simple workflow. While saving the workflow diagram, the Workflow Title dialog is displayed for the name of the workflow, as shown in Figure E2.

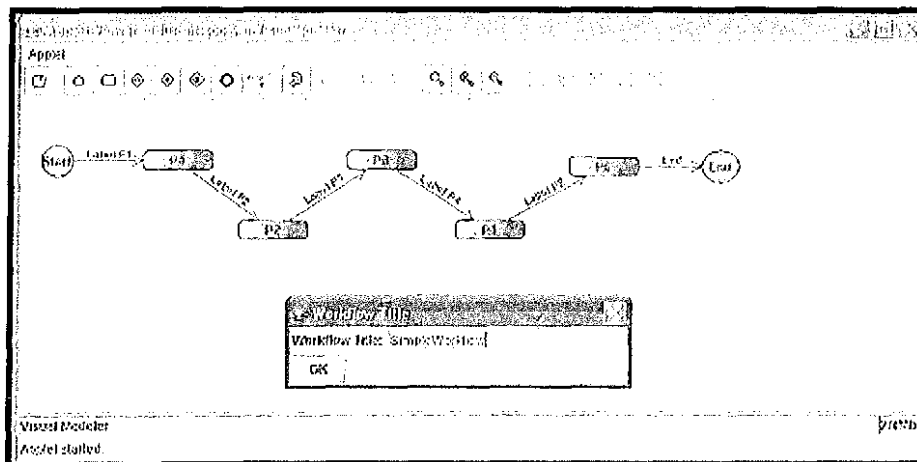
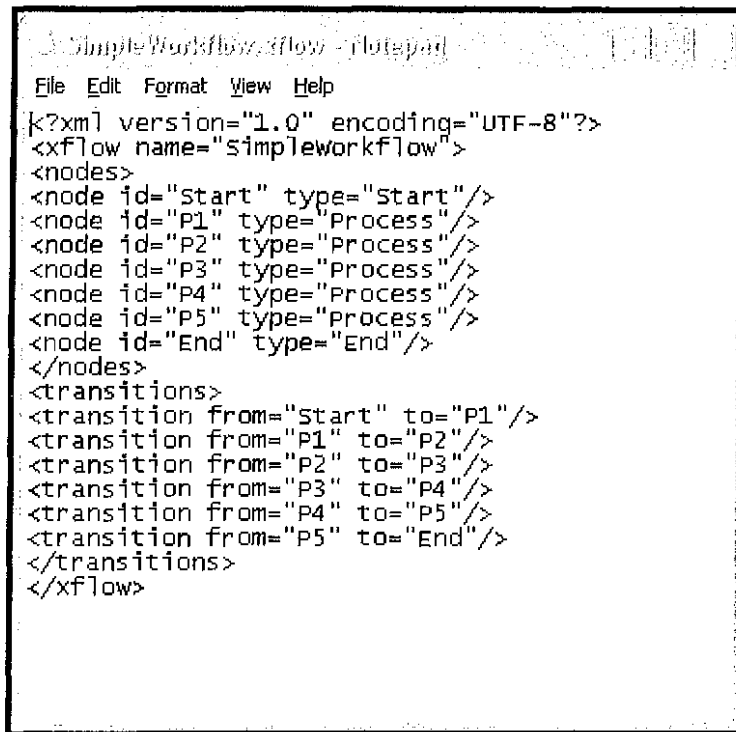


Figure E2: Workflow Title Dialog of VM for XFlow

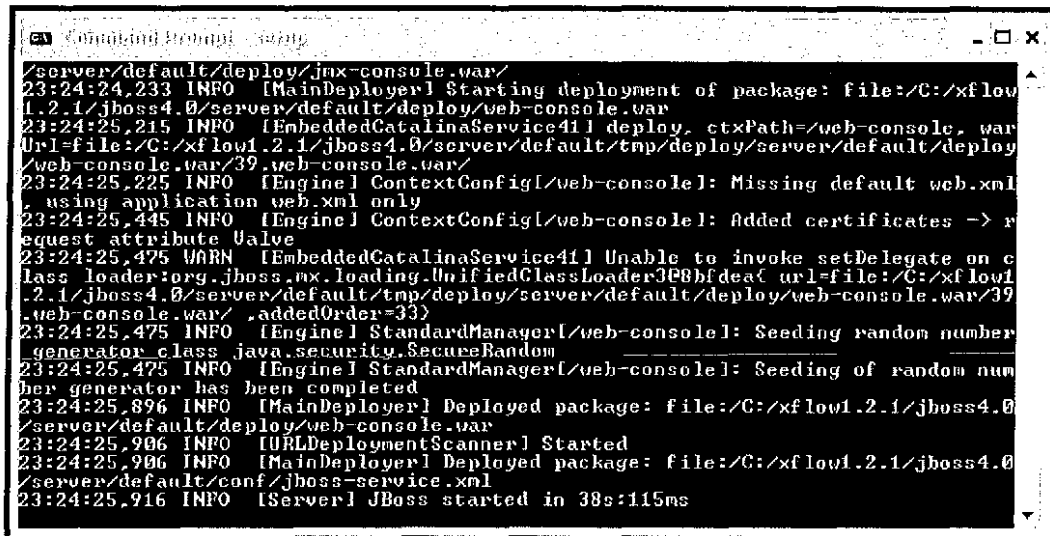
After the workflow is saved, a XFLOW format workflow file is generated in the XFlow applet folder. The file is shown in Figure E3.

A screenshot of a Notepad window titled "SimpleWorkflow.xflow - Notepad". The window contains XML code for a workflow. The code starts with a root element <xflow name="simpleworkflow">. Inside, there are two main sections: <nodes> and <transitions>. The <nodes> section lists six nodes: "Start" (type="Start"), "P1" (type="Process"), "P2" (type="Process"), "P3" (type="Process"), "P4" (type="Process"), "P5" (type="Process"), and "End" (type="End"). The <transitions> section lists five transitions: from "Start" to "P1", from "P1" to "P2", from "P2" to "P3", from "P3" to "P4", from "P4" to "P5", and from "P5" to "End". The XML code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xflow name="simpleworkflow">
  <nodes>
    <node id="Start" type="Start"/>
    <node id="P1" type="Process"/>
    <node id="P2" type="Process"/>
    <node id="P3" type="Process"/>
    <node id="P4" type="Process"/>
    <node id="P5" type="Process"/>
    <node id="End" type="End"/>
  </nodes>
  <transitions>
    <transition from="Start" to="P1"/>
    <transition from="P1" to="P2"/>
    <transition from="P2" to="P3"/>
    <transition from="P3" to="P4"/>
    <transition from="P4" to="P5"/>
    <transition from="P5" to="End"/>
  </transitions>
</xflow>
```

Figure E3: SimpleWorkflow.xflow Output File

In a command window, the user starts the JBoss Server, using the startup scripts distributed with the XFlow package. The output of the console window after the JBoss Server startup is shown in Figure E4.



```

C:\> cd /d C:\xflow\1.2.1\jboss4.0\server\default\deploy\jmx-console.war/
23:24:24.233 INFO [MainDeployer] Starting deployment of package: file:/C:/xflow
1.2.1/jboss4.0/server/default/deploy/web-console.war
23:24:25.215 INFO [EmbeddedCatalinaService41] deploy, ctxPath=/web-console.war
url=file:/C:/xflow1.2.1/jboss4.0/server/default/tmp/deploy/server/default/deploy
/web-console.war/39.web-console.war/
23:24:25.225 INFO [Engine] ContextConfig[/web-console]: Missing default web.xml
, using application web.xml only
23:24:25.445 INFO [Engine] ContextConfig[/web-console]: Added certificates -> r
equest attribute Valve
23:24:25.475 WARN [EmbeddedCatalinaService41] Unable to invoke setDelegate on c
lass loader:org.jboss.mx.loading.UnifiedClassLoaderJPEBbfdea url=file:/C:/xflow1
.2.1/jboss4.0/server/default/tmp/deploy/server/default/deploy/web-console.war/39
.web-console.war/.addedOrder=33)
23:24:25.475 INFO [Engine] StandardManager[/web-console]: Seeding random number
generator class java.security.SecureRandom
23:24:25.475 INFO [Engine] StandardManager[/web-console]: Seeding of random num
ber generator has been completed
23:24:25.896 INFO [MainDeployer] Deployed package: file:/C:/xflow1.2.1/jboss4.0
/server/default/deploy/web-console.war
23:24:25.906 INFO [URLDeploymentScanner] Started
23:24:25.906 INFO [MainDeployer] Deployed package: file:/C:/xflow1.2.1/jboss4.0
/server/default/conf/jboss-service.xml
23:24:25.916 INFO [Server] JBoss started in 38s:115ms

```

Figure E4: JBoss Server Startup Trace Messages

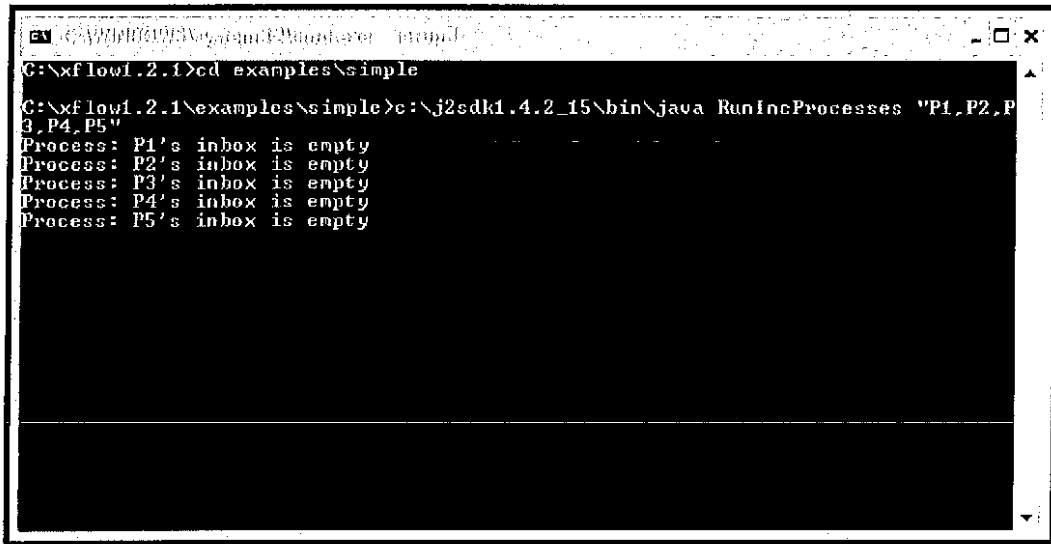
In another command window, the user deploys the SimpleWorkflow, using the DeployModel tool, as shown in Figure E5.



```
C:\xflow\2.1>cd c:\xflow\2.1\examples\simple
C:\xflow\2.1\examples\simple>c:\jdk1.4.2_15\bin\java xflow.tools.DeployModel
SimpleWorkflow.xflow
Success
C:\xflow\2.1\examples\simple>
```

Figure E5: Deploy SimpleWorkflow using DeployModel

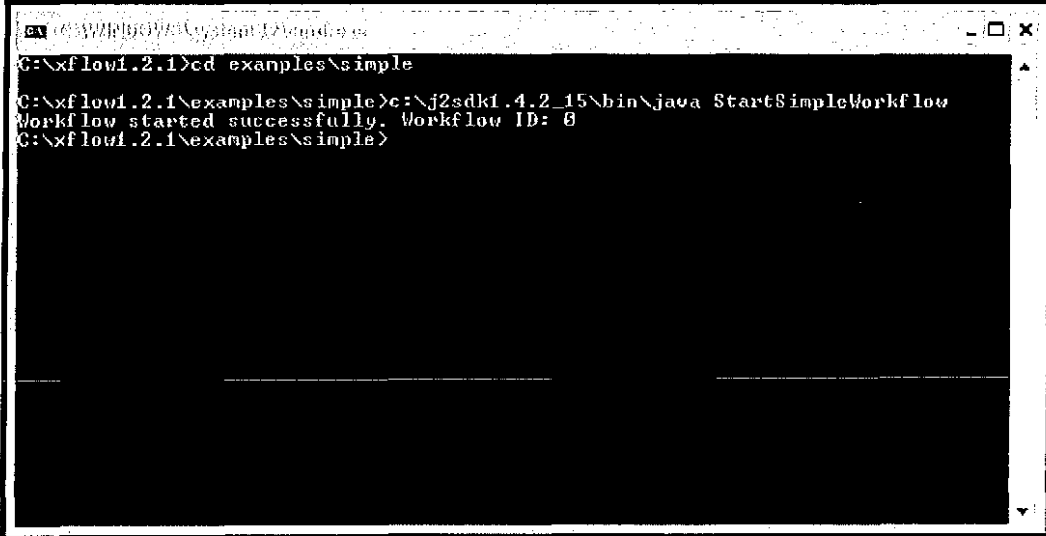
In another command window, the user runs the Increment Processes for all the processes, as shown in Figure E6.



```
C:\xflow1.2.1>cd examples\simple
C:\xflow1.2.1\examples\simple>c:\j2sdk1.4.2_15\bin\java RunIncProcesses "P1,P2,P3,P4,P5"
Process: P1's inbox is empty
Process: P2's inbox is empty
Process: P3's inbox is empty
Process: P4's inbox is empty
Process: P5's inbox is empty
```

Figure E6: Increment Process Started

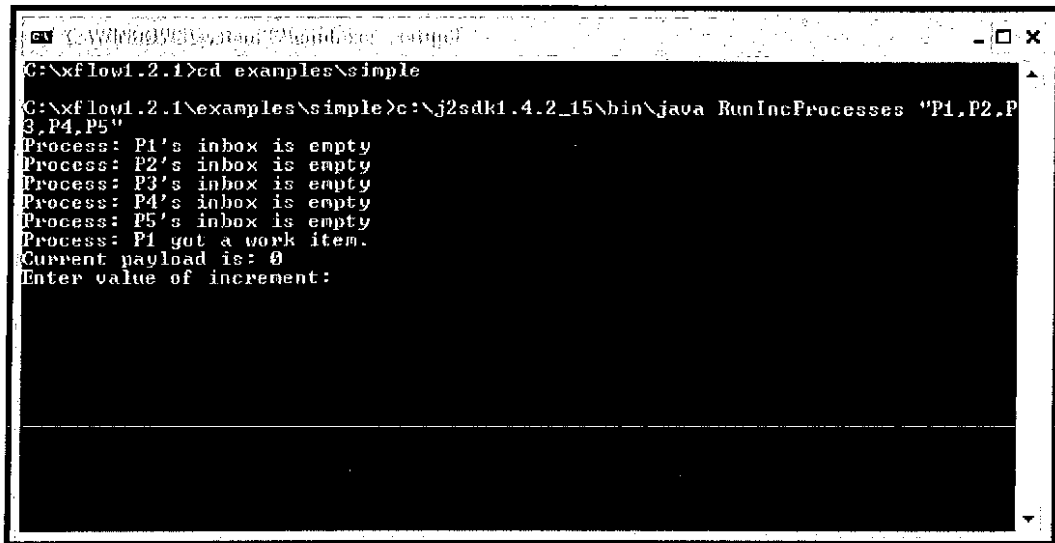
In yet another command window, the user now creates an instance of SimpleWorkflow. Figure E7 shows the workflow started with a new Workflow ID.



```
C:\xflow1.2.1>cd examples\simple
C:\xflow1.2.1\examples\simple>c:\j2sdk1.4.2_15\bin\java StartSimpleWorkflow
Workflow started successfully. Workflow ID: 0
C:\xflow1.2.1\examples\simple>
```

Figure E7: Workflow Instance Created

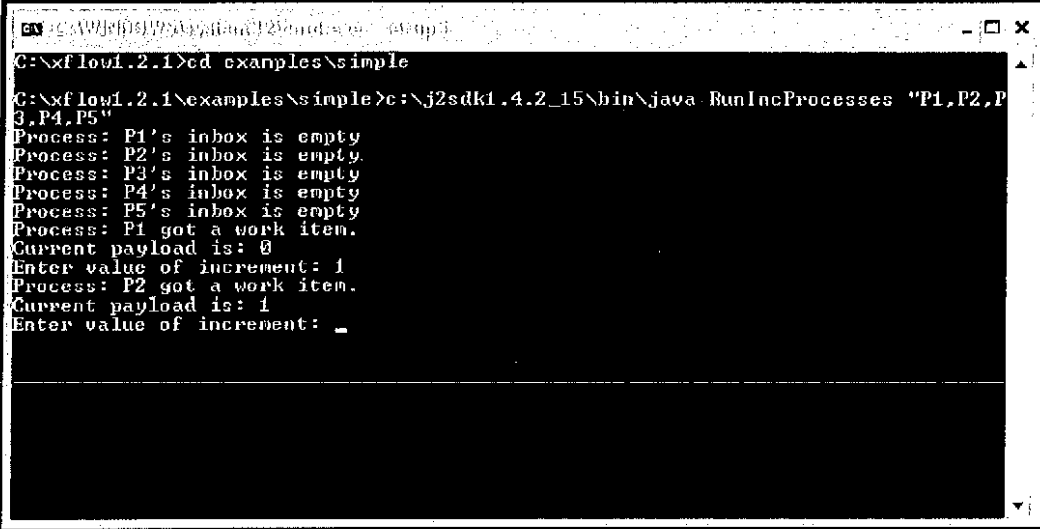
The first process of SimpleWorkflow then receives the work item and waits for a user response, as shown in Figure E8.



```
C:\xflow1.2.1>cd examples\simple
C:\xflow1.2.1\examples\simple>c:\j2sdk1.4.2_15\bin\java RunIncProcesses "P1,P2,P3,P4,P5"
Process: P1's inbox is empty
Process: P2's inbox is empty
Process: P3's inbox is empty
Process: P4's inbox is empty
Process: P5's inbox is empty
Process: P1 got a work item.
Current payload is: 0
Enter value of increment:
```

Figure E8: Workflow Process Receives a Work Item

The SimpleWorkflow can be seen in action, when the second process receives the work item and waits for a user response, as shown in Figure E9.



```
C:\xf low1.2.1>cd examples\simple
C:\xf low1.2.1\examples\simple>c:\j2sdk1.4.2_15\bin\java RunIncProcesses "P1,P2,P3,P4,P5"
Process: P1's inbox is empty
Process: P2's inbox is empty
Process: P3's inbox is empty
Process: P4's inbox is empty
Process: P5's inbox is empty
Process: P1 got a work item.
Current payload is: 0
Enter value of increment: 1
Process: P2 got a work item.
Current payload is: 1
Enter value of increment: _
```

Figure E9: SimpleWorkflow in Action

VITA

Jyoti Chaturvedi received a Bachelor of Engineering degree with a major in Electrical Engineering from the Government Engineering College, Ujjain, MP, India, and a Master of Business Administration degree with a major in Marketing from Jawaharlal Nehru Institute of Business Management, Ujjain, MP, India. She expects to receive Master of Science degree in Computer and Information Sciences from the University of North Florida in May 2008. Dr. Arturo Sánchez-Ruíz of the University of North Florida is serving as Jyoti's thesis advisor.

Jyoti currently works as a database Web specialist for the Florida Virtual School. Prior to this, she was employed as a marketing executive at Biochem Synergy, Ltd., India; a visiting faculty member at Prestige Institute of Management and Research, India; a software developer at Harbour Management Consultants, Bedford, MA; a systems analyst at Amdocs, Inc., Champaign, IL; and a software engineer at Adtec Digital, Jacksonville, FL.