Kumaraguru College of Technology

Department of Computer Science and Engineering
Coimbatore– 641006.
April 2003

FLOWMATE WITH TAMINO

Project work done at

P-1018

SRA Systems Ltd., Chennai.

PROJECT REPORT

Submitted in partial fulfillment of the
Requirements for the award of the degree of

Master of Computer Applications

Bharathiar University, Coimbatore

Submitted by

RAMPRASAD G
Reg.No: 0038M1059

Internal Guide

Mr. S. Ganesh Babu., M.C.A.,
Dept. of Computer Science & Engineering,
Kumaraguru College of Technology,
Coimbatore

External Guide

Mr. Ajit Kumar Maddali
Assistant Manager (Projects)
SRA Systems Ltd.,
Chennai

CERTIFICATE

This is to certify that the project work entitled

FLOWMATE WITH TAMINO

Submitted to the

Department of Computer Science and Engineering

Kumaraguru College of Technology

In partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of original work done by Ramprasad G, Reg.No.0038M1059 during his period of study in the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore under my supervision and this project work has not formed the basis of award of any Degree/Diploma Associateship/Fellowship or similar title to any candidate of any university.

[illegible]
Professor and Head

[illegible]
Staff-in-charge

Submitted for University Examination held on 16.04.2003

[illegible] 16/04/03
Internal Examiner

[illegible]
External Examiner

Meeting Demands World Wide

SRA Systems Limited

March 17, 2003

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Ramprasad G** who is pursuing his final year M.C.A (Master of Computer Applications) at Kumaraguru College of Technology, Coimbatore has successfully completed his project work entitled **"FLOWMATE WITH TAMINO XML SERVER"** as part of his academic curriculum during December 2002 and March 2003.

During the project tenure his attendance was found to be regular.

Due to operational reasons, we are not in a position to give the source code to the student.

We wish him all the best in his future endeavours

[illegible]

Ajit Kumar Maddali
Assistant Manager - Projects

DECLARATION

I hereby declare that the project work, 'FLOWMATE WITH TAMINO ' submitted by me (Ramprasad G, 0038M1059) towards the fulfillment of the degree of Master of Computer Applications from Bharathiar University has not formed the basis for the award of any degree, diploma or association of similar titles. The project work is done independently by me under the guidance of Mr.S.Ganesh Babu (Internal Guide) and Mr.Ajitkumar Maddali (External Guide).

[illegible]

Ramprasad G

[illegible]

Internal Guide

ACKNOWLEDGEMENT

Before I present this project, I wish to express my sincere gratitude to **Dr.K.K.Padmanabhan, Ph.D.,** Principal, Kumaraguru College of Technology, Coimbatore for his encouragement.

I wish to express my sincere thanks to the head of the department **Dr.S.Thangasamy, Ph.D.,** for having given me permission to carry out the project.

I also owe a deep sense of gratitude to **Mr.Ajitkumar Maddlai** Assistant, Manager (Projects), **Mr.Chandrasekaran** Senior Associate for their Guidance, Constructive ideas and Constant Inspiration throughout the duration of the project.

Special thanks are due to **Mr.S.Ganesh Babu, M.C.A.,** my guide and tutor for his continued interest and guidance at all stages of this project.

Last in words but first in thoughts I owe my thanks to my family and friends for putting up with my strange ways and occasionally short temper for all these days.I thank all those who have contributed in various ways to this project by ways of their valuable suggestions and ideas.

SYNOPSIS

FlowMate, a generic WorkFlow Management Product is being developed by SRA. It can be applied to any sector with minor customization according to the client's business logic. This product uses an Oracle Database Server to store WorkFlow related data. All data exchange among the components of Flowmate is in the XML format. With an Oracle server, storing such Hierarchical data is not feasible since it was designed with the Relational Model in mind. When XML documents have to be stored it requires essentially a tree structure. Thus there is an impedance mismatch here.

XML files had to be stored in BLOB (Binary Large Objects) in Oracle. But retrieving such data in XML format is a tedious task and requires processing to convert the BLOB to XML format. The efficiency of the system is degrades due to this overhead. Therefore a permanent solution has to be found wherein this impedance mismatch is resolved.

The team proposes to use a Tamino XML Server to this end. FlowMate integrated with Tamino will break free from the shortcomings of using an RDBMS. For this migration to happen, the first step is design equivalent schemas, which correspond to the existing table structure, has to be designed. Next, the data access layer & all components, which access it are redesigned & re-written. Implementation & Testing is carried out in detail so that the newly integrated system works perfectly.

TABLE OF CONTENTS

INTRODUCTION

1.1.ABOUT THE PROJECT

WORKFLOW

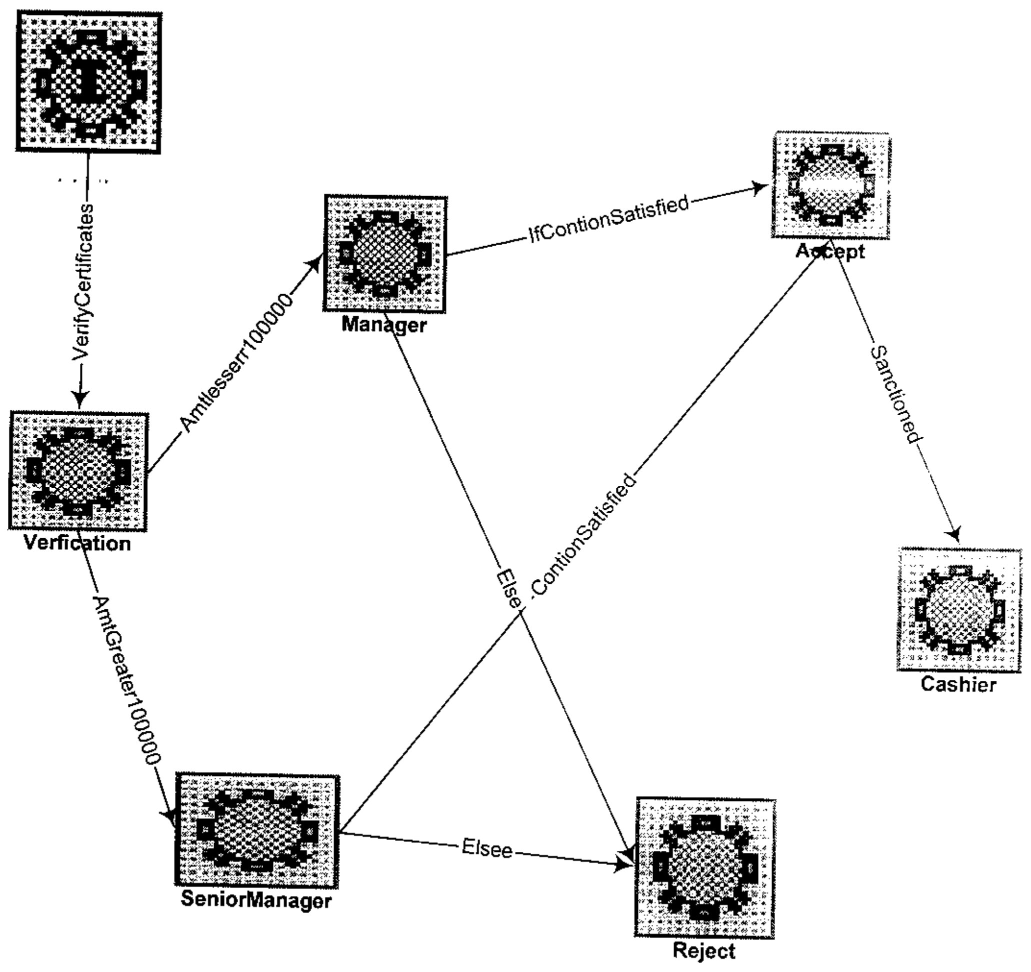
Workflow can be described simply as the movement of documents and tasks through a business process. The business processes make up the 'infrastructure' of a company. Workflow can be a sequential progression of work activities or a complex set of processes each taking place concurrently according to a set of rules.

A workflow essentially consists of stages, stage inputs and rules pertaining to a stage. The work progresses from one stage to another to completion based on rules. Workflow helps you:

- Streamline your business processes.
- Decrease cost of business processes through automation.
- Speed up business processes and deliver faster service to your customer.
- Track and control business processes

Let us consider a real time scenario of a Loan Application WorkFlow:

LOAN APPLICATION WORKFLOW



The input stage is one that gets all the facts from the user. The information is then passed to the verification stage. The verification stage has a set of rules regarding the dimension of the amount.

In the verification stage, depending on the size of the amount, they are passed to the respective stages. If the amount is greater than one lakh, the sanctioning of the loan is being done by the Senior Manger, and if the amount is less than one lakh, the sanctioning can be done by the Branch Manager.

The Senior Manger and the Branch Manager, after verifying all the essentials regarding the loan, can either approve the loan or reject the loan. If the loan is sanctioned, the details regarding the loan are sent to the cashier.

FLOWMATE AND ORACLE

FlowMate is a workflow solution developed by SRA Systems. FlowMate enables automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant (human or machine) to another for action, according to a set of procedural rules. FlowMate is a web-enabled, workflow enabling product framework. FlowMate is a front-end application that integrates with ORACLE to store and retrieve the workflow data.

TAMINO XML SERVER

The **T**ransaction **A**rchitecture for the **M**anagement of **I**nternet **O**bjects (TAMINO) is a native XML repository developed by SoftwareAG. It is capable of storing XML documents natively without transformation to other formats. The Tamino XML Server is an information server that uses XML to store XML data from multiple sources in a reliable manner and then to exploit XML's transformation capabilities to publish the stored data in a variety of formats, allowing the XML repository to be used for the full range of devices in use today and in the future. Tamino therefore meets the requirement for a common access and exchange mechanism for organizations that have already or are planning to implement web-based applications on the basis of existing data. It integrates cleanly with web servers and application servers in existing infrastructures. It is a totally new data management concept.

Features of TAMINO:

- Built-in native-XML data store for fastest DB access.
- Storage and retrieval of XML data via HTTP & TCP/IP.
- Connection to standard Web servers.
- Integration existing DBMS sources.
- Interfacing with XML authoring, schema, and other XML tools.
- Based on XML standards.

FMTAMINO

The requirements have been identified to integrate FlowMate with TAMINO for optimal performance. It has been proposed to identify those components of FlowMate, which interact with ORACLE and modify them to interact with TAMINO. The projects involves working with one such component called the Workflow Definer Tool (WDT), which is used for defining a workflow, and bring forth the changes necessary for it to integrate with TAMINO.

1.2 PLAN OF REPORT

Selection of Organization

This chapter focuses on the profile of SRA Systems. It touches upon its Origin, Growth, Business Focus, Technology Focus, Domain, Infrastructure and Resources.

Background Study

The Background study involves research on the existing system (FlowMate with ORACLE), the drawbacks of such integration and the proposed system (FlowMate with TAMINO XML Server - FMTAMINO).

Problem Formulation

The Problem Formulation encompasses the main and the specific objective of the project and also the platform selected for implementation. The main objective focuses on the implementation of the proposed system. The specific objective focuses on modifying the identified component.

System Analysis and Design (High-Level)

This chapter details on the architecture of FlowMate and TAMINO and the architecture design of the proposed system.

System Analysis and Design (Low-Level)

The Low-Level design identifies the components of FlowMate that need to be modified. A detailed diagrammatic description of WDT is given here along with the Code design and the likes.

Testing and Implementation

This chapter gives the description of the tests carried out to establish the correctness of the module. The implementation issues concerning FMTAMINO are also covered in this chapter.

Scope for further development

This chapter talks about the various ways by which the system can be enhanced in the future.

SELECTION OF ORGANIZATION

SRA Systems Limited is a quality conscious international IT organization based at Chennai, India. Founded and promoted by a group of technocrats in 1986, supported through subsequent equity investments by multinationals, SRA has been successfully implementing projects in the US, Europe, Japan and the Middle East. SRA has significant direct presence in USA to enter to the North American market. SRA has ISO 9001 certification and recently assessed for *SEI CMM Level 4.*

SRA addresses IT opportunities in India, North America, Europe, Japan, Malaysia and Australia. SRA carries out large onsite and offsite projects. Clients in the US, Europe and Japan have chosen SRA as their Offshore Development Center (ODC).

Business Focus

SRA's business focus is on Custom Software Services, carried out onsite, offsite and offshore. Using a true 'Global Delivery Model', SRA delivers good business advantages to the clients.

LINES OF BUSINESS
• Custom Software Development • Re-Engineering • Maintenance Services • Support Services

Custom Software Development

Custom Development of new products and applications is a major focus area of SRA and this activity spans across several Industry verticals. SRA's Custom Software Development Services include developing, designing and providing end-to-end custom solutions. SRA has developed applications/products in the following areas:

- Applications in functional areas like Finance, Human Resources etc
- End-to-End retail applications
- Total Turnkey MRP solutions

- Back Office Systems with Imaging & Workflow capabilities
- Document Management Solutions
- Natural Language Processing Solutions
- Embedded systems
- Testing solutions for semi-conductor industry

Re-Engineering

SRA has over 500-person years experience in executing migration and re-engineering projects. Elements of this experience includes

- Wide cross platform skills
- In-house developed conversion tools and utilities
- Expertise in deploying commercially available off the shelf conversion tools
- Metrics driven project management experience

SRA has formalized this experience into a structured methodology called RENEW (Re-Engineering & Migration Execution Work plan) and successfully deployed this in several large projects. RENEW has a unique approach for migration and re-engineering projects. The RENEW methodology is platform independent and could be deployed for IBM mainframe and open systems based projects.

RENEW addresses the following types of re-engineering and migration projects:

- Conversion of applications on old version to a new version
- CICS conversion - from macro based to command based
- Language conversion – from PL/1 to C or COBOL, MVS Assembler to C/COBOL
- Database Migrations
- Web enabling existing applications
- Porting of Document management system on OS/390 and web enabling it.
- Porting applications from other mainframes / UNIX /Windows/Open Systems to IBM OS/390
- Porting from IBM non MVS platforms to OS/390
- Porting to Windows, Windows NT and different flavors of Unix
- Language conversion from Legacy to Open Systems

- Database Conversions from Legacy to Open Systems
- Application conversions from 2 tier to n tier
- Architecture conversions from proprietary to structured/object oriented/distributed.

Maintenance Services

Organizations all over the world are realizing the cost benefit opportunities of outsourcing their software maintenance jobs. SRA offers comprehensive maintenance solutions in various industries. The solutions include

- Long term Maintenance solutions for Products and Applications developed by SRA
- Maintenance solutions for software developed by clients and third parties
- Maintenance cum software enhancement solutions for both SRA developed software and Non-SRA developed software.

SRA has considerable amount of experience in maintaining software ranging from legacy software to open systems to specialized embedded solutions.

Support Services

Support Services enables SRA to offer an almost one-stop Solution to the clients' custom service requirements. The support services from SRA include:

- Testing Services: Application and product testing services in special customized test labs that use both SRA developed tools and off-the shelf products.
- Transcription Services: Data conversion from one format to another, capture of document data, conversion of design data and digitization of maps etc.

Technology Focus

SRA derives sustained competitive advantage through the high technology content in most of the projects carried out by SRA. The company has a separate R&D team, which constantly looks out for future technology growth areas and suitably builds up expertise in those areas. Training and updating skills on technology have contributed to SRA's growth over the past few years. SRA's farsightedness in identifying future technology areas helps in expanding its business and also reduces the lead-time in coping up with the newer technologies.

TECHNOLOGY FOCUS
• Internet Solutions • Document Management Solutions • Knowledge Management Solutions • Mobile Computing • Embedded Systems Solutions

Infrastructure and Resources

SRA's development center is situated at Chennai. Sales and technical offices are located at USA, UK, Japan, Malaysia and Australia. SRA has fully owned subsidiary – Software Systems and Solutions (3S) Inc. in USA. 3S has offices at Atlanta – Georgia; Chicago – Illinois; Dallas-Texas; Denver-Colorado; Santa Clara-California. SRA has offices at Middlesex, UK to cater to the European market, Kyoto in Japan and Sydney in Australia.

Skills	
Operating Systems	Various UNIX flavors, WIN NT, MVS and Win 9x.
Databases	Oracle, Sybase, SQL Server, DB2, Ms Access, SQL Anywhere, Tamino.
Programming Languages	C, C++, COBOL, PL/I.
TP Environment	Tuxedo, CICS, MTS.
Tools	
Front-End	Developer 2000, Power Builder, VC++, VB. and Designer 2000.
Web Based	Active X, Java, VJ++, Visual Café, Visual Studio, Cold-Fusion and ASP.
Data Modeling	E-R Win, S-Designer.
Object Modeling	Rational Rose.
Testing and CM	SQA Robot, Visual test, Visual Source Safe, Clear Case.
Web Servers	Apache, Netscape, IIS, and IPlanet.

Application Servers	Broad Vision, BEA Web logic, Interwoven.
Others	MSDN, Robohelp, Macromedia Products.

SRA uses high-speed Internet links, which support communication for offshore development. Dedicated links will be established with customers based on requirement.

Quality Systems

"We are committed to Continuous Improvement of quality of Products and Customer services by adhering to International Standards."

The above stated Quality Policy of SRA is practiced in the daily activities at SRA. SRA looks at Quality Management as a route to higher productivity, better customer satisfaction and personnel enrichment.

SRA's major focus areas in the Quality Management are

- Software Engineering
- Estimation
- Requirements Management
- Testing
- Metrics

SRA strongly believes that it is not enough to have a sound requirements analysis but managing requirements throughout the project's life span is more important to control the schedules and costs. Estimation strategy will provide the basis for controlling requirements. Metrics collection is given high importance in SRA. Every task gets subjected to measurement.

BACKGROUND STUDY

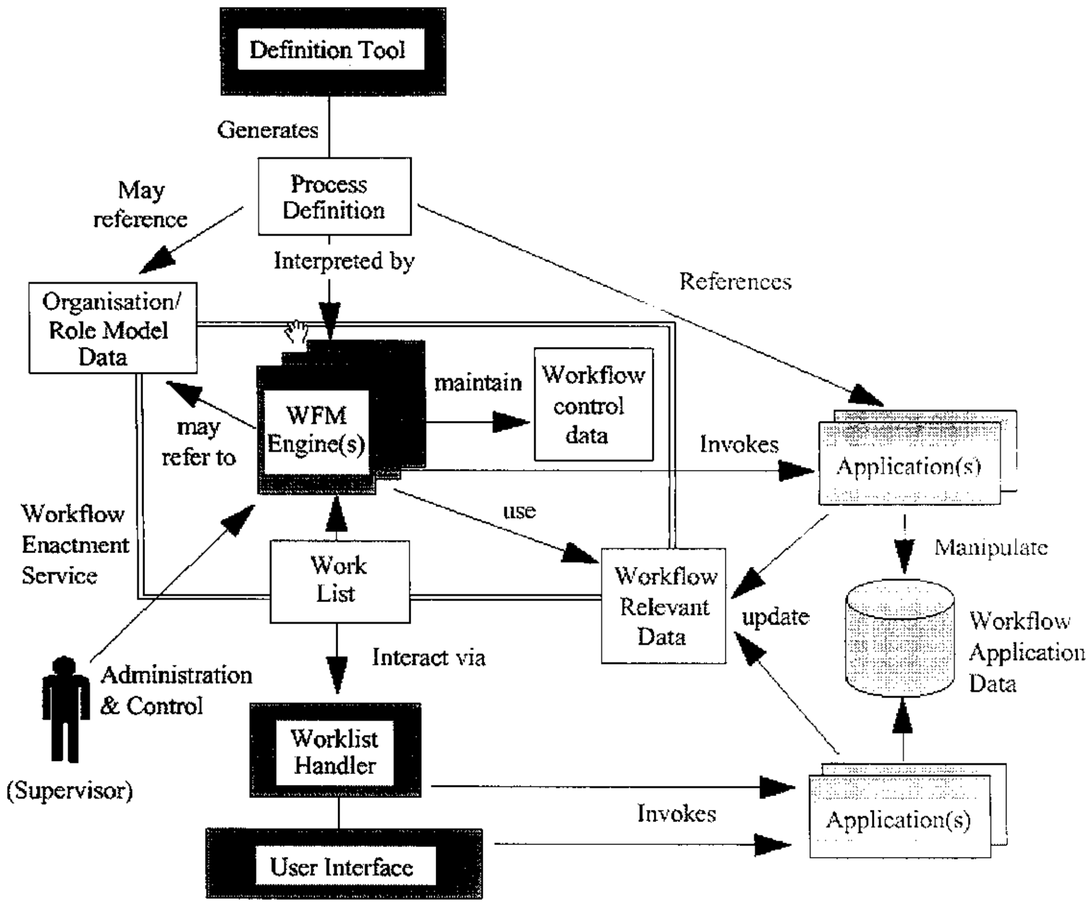
3.1 Existing System:

FlowMate is currently integrated with **Oracle 8.1.3.7 Database Server**. Data exchange between the components of FlowMate is in XML format. XML is being used extensively in FlowMate in the following areas.

a. Storage of Workflow definition
b. Storage of Business process rules
c. Exchange of information between Client tier to Middle tier
d. Exchange of information between Middle tier to backend server
e. Rendering of information at the client end is being done using XSL.

As ORACLE handles only relational data, the XML data is converted to a form compatible for storage and retrieval.

FlowMate Architecture



Architecture Description

The application essentially uses the *Workflow Management Coalition Specification (WFMC)* Architecture.

The Main functional components of the FlowMate system are illustrated in the above figure. This model has three types of component:

Software components, which provide support for various functions within the workflow system, are called the Workflow Engine (WFE in FlowMate).

Various types of System definition and control data, which are used by one or more software components.

Applications and Application databases which are part of the FlowMate product but which can be invoked by the total Workflow system.

3.2 Limitations of Existing Database:

There are a lot of drawbacks in using Oracle as the backend for FlowMate. It's limitations come into light when the following functions are performed with respect to FlowMate:

✓ **Storage**

The Input Information captured by the WDT & ODT is in the form of XML initially. This XML information is converted into the rigid flat file structure of relational data model, this transformation results in the classical impedance mismatch problem.

✓ **Retrieval**

While retrieving data from Oracle, the conversion procedures implemented have to regenerate the exact replica of the shredded XML document. The process of regenerating the hierarchical data model from a flat model usually results in information loss, which is otherwise called the "Round Trip loss".

It is here that the need for a native XML repository is desperately felt. Since the XML files are stored in a relational format, querying them is no easy task.

✓ **Table Design**

The table design in the relational formats is rigid. For e.g., a record has to confirm to all fields of the table. XML does not adhere to this Rule, which at times & specifically in the FlowMate context can be a bane rather than a boon.

3.3 Proposed System

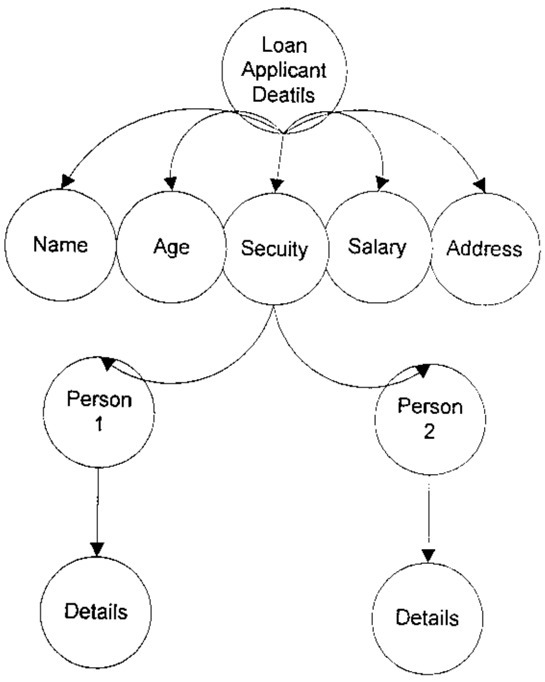
The Solution to the Drawbacks mentioned in the previous section is to use a Native XML Repository. Hence Software AG's Tamino was selected to replace Oracle in FlowMate Software AG's Tamino Database is a **Native XML Database**; it is essentially suited for storing and retrieving XML data natively. In essence, the Tamino XML Server is an information server that uses XML to store XML data from multiple sources in a reliable manner and then to exploit XML's transformation capabilities to publish the stored data in a variety of formats, allowing the XML repository to be used for the full range of devices in use today and in the future. The basic advantage Tamino has is that it stores, queries and transforms XML data *natively*, using the Web and XML specifications and interfaces. It integrates cleanly with web servers and application servers in existing infrastructures.

Approach:

The idea is to integrate FlowMate with Tamino and will be known as FMTamino. This will happen via Tamino API's. The implementation of APIs is going to happen for the modules of FlowMate, one of which is the WDT. The WDT is the component, which captures the WORKFLOW in the XML format. Since the output of WDT is in XML data, the data cannot be stored into the Oracle database natively. So the output of this component is converted from XML format to relational data and then stored in Oracle Database. While retrieving the data from the database, it is converted into XML and then is handled by the WDT.

For integration with Tamino, the Data Access Layer of WDT is replaced with a set of APIs, which will perform the function of connecting FlowMate with Tamino. Thereafter XML data can be stored and retrieved natively which considerably reduces the overhead. The XML instance stored in Tamino by the WDT is then utilized by the Engine to dynamically create the client application. The above example of the loan processing system when stored will be a tree structure in Hierarchical Model.

Hierarchical



The entire project is divided into the following phases:

Phase I	- Identification of existing components that form the data access layer.
Phase II	- Arrive at a schema to migrate the RDBMS structure to XML schema.
Phase III	- Re-engineer the Database layer with Tamino connectors to seamlessly connect to Tamino repository.
Phase IV	- Integrate with existing system.
Phase V	- Test and implement the system.

Phase I:

WFMAT is the component, which is used by the WDT to interact with Oracle for the various functions like adding, modifying & deleting records & also for retrieving values. It comprises of two Classes

ClsATFunction

ClsDatabase.

Phase II:

The WDT is the component where the information about the workflow is captured. It stores the information in 22 tables that are listed below.

Table Name

- WFMWorkFlow
- WFMStage
- WFMStageInputs
- WFMRule
- WFMRuleStage
- WFMRuleStageOutput
- WFMRuleStageOutputMultiple
- WFMListValues
- WFMStagedDocuments
- WFMGlobalVariables
- WFMDeadLine
- WFMConfig
- WFMGroupOrder
- WFMStyleSheets
- WFM2Dstrucutre
- WFM2Ddetails
- WFMStage2Dstructure
- WFMActionType

- WFMStageActions
- WFMStageActionOutput
- WFMStageActionInputs
- WFMStage2Ddetails

In our attempt to model the information in the Oracle Tables to Schemas in Tamino, the first thing that we have to remember is that the rules which hold good for Oracle do not hold true for Tamino.

The rules, which we stuck to while designing, the schema's are:

- Business Objects will be modeled into in an independent schema. The above statement is true because the information modeled in the Business Object can exist independently or in conjunction with other Business Objects
- The lesser the joins, the lesser the complexity & faster the query access.

The Schemas were designed having these concepts in mind.

Phase III:

The WFMAT component, which is Data Access Layer for WDT, has to be re-engineered, due to fact mentioned earlier that the rules, which hold good in Oracle, are not applicable for Tamino. As such the Queries written in SQL & embedded in WFMAT have to be changed to XQL (XML Query Language). The component has to be re-written so that it addresses the needs for communicating with Tamino.

Phase IV:

Before the WFMAT component is integrated with the existing system, it has to be verified whether all the needs of FlowMate are addressed. A copy of the data in Oracle is backed up. It is then migrated to Tamino & checked if all components addressing the Data Access layer work perfectly, if not better.

Phase V:

A detailed note of all possible functionality of FlowMate, which uses Oracle, was made before its replacement with Tamino. Tests were done to make sure that previous functionality was possible in FlowMate with Tamino. Also the Schemas were cross verified to make sure that they comply with the cardinality & enumeration constraints set in the Oracle Tables.

PROBLEM FORMULATION

4.1 THE PROBLEM

The idea of the using Tamino XML server as a Back End for FlowMate is to decrease processing time and also increase the efficiency of using XML. The data that is captured form the Workflow Definition Tool consists of XML data, and it would be easy to store the XML files natively in Tamino. The captured XML files are stored in Oracle by converting them in to Binary Large Data Objects (BLOBs). Since conversion of data is done, in the retrieval process, the files are to be converted back again.

Impedance Mismatch

In FlowMate the XML data captured is ripped and stored in tables in Oracle. Data of a tree-structured model is made to fit into relational model. This leads to impedance mismatch as the relational data model is flat and rigid structured while the XML data model is nested. The 2 data models being incompatible result in an overhead in the form of ripping XML documents while storage.

Round Tripping

The ripped XML documents during retrieval are regenerated. There is no certainty whether it is possible to regenerate the exact replica of the document ripped and stored. Round Tripping results in unnecessary overhead of format conversion during document publishing.

Rigid, Fixed Structure

The Relational model enforces a rigid and fixed structure, which is incapable of handling loose and flexible nature of XML documents.

Storage Space

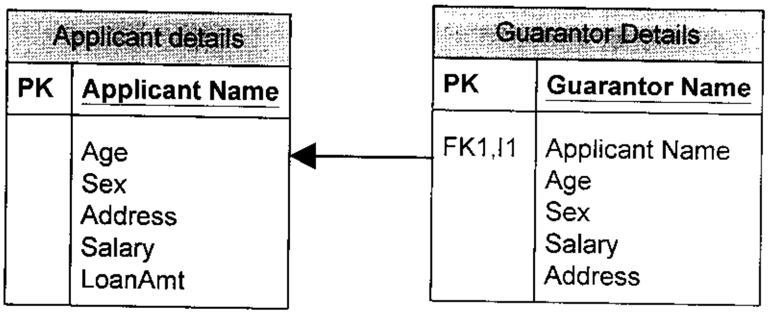
While storing XML data in tables, unnecessary overhead of storage space is encountered. Shredding an XML document in a relational database occupies 2.5 times mores storage space.

Query Capabilities

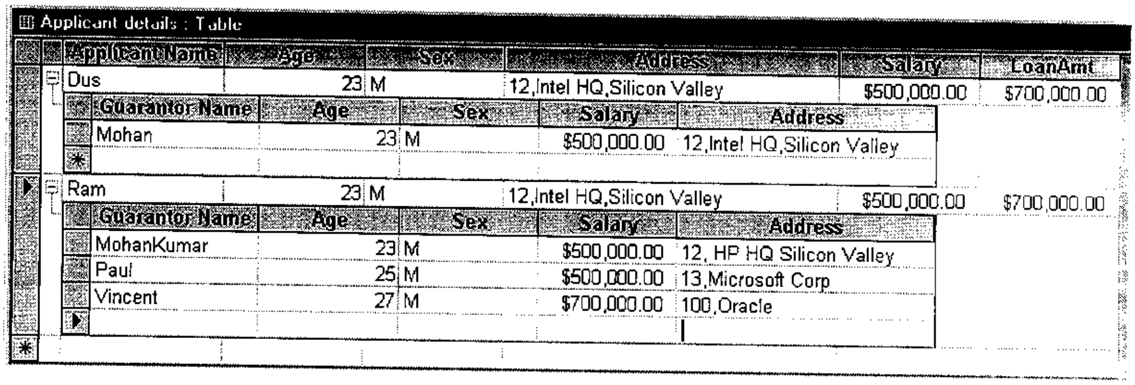
The querying capabilities of XML are not fully exploited in the Relational setup.

The Loan Processing Workflow illustrated in previous sections when stored in the Relational Model is stored in this format. The Relational storage has to have many joins (P Key-F Key) which makes the structure unwieldy & slow in terms of retrieval & access. . The Applicant details & the Guarantor Details are stored in two separate tables. There is a separate record in the guarantor table for every guarantor the Applicant provides.

TABLE STRUCTURE



DATA STORED IN RELATIONAL MODEL



Applicant details : Table

Applicant Name	Age	Sex	Address	Salary	LoanAmt
Dus	23	M	12,Intel HQ,Silicon Valley	$500,000.00	$700,000.00
Ram	23	M	12,Intel HQ,Silicon Valley	$500,000.00	$700,000.00

Guarantors of Dus:

Guarantor Name	Age	Sex	Salary	Address
Mohan	23	M	$500,000.00	12,Intel HQ,Silicon Valley

Guarantors of Ram:

Guarantor Name	Age	Sex	Salary	Address
MohanKumar	23	M	$500,000.00	12, HP HQ Silicon Valley
Paul	25	M	$500,000.00	13,Microsoft Corp
Vincent	27	M	$700,000.00	100,Oracle

DATA STORED IN HIERARCHICAL MODEL

```
- <Applicant>
    <Name>Ram</Name>
    <Age>23</Age>
    <Sex>M</Sex>
    <Address>12,Intel HQ,Silicon Valley</Address>
    <Salary>$500,000.00</Salary>
    <LoanAmt>$700,000.00</LoanAmt>
  - <Guarantor>
      <Name>Mohan Kumar</Name>
      <Age>23</Age>
      <Sex>M</Sex>
      <Address>12, HP HQ Silicon Valley</Address>
      <Salary>$500,000.00</Salary>
    </Guarantor>
  - <Guarantor>
      <Name>Paul</Name>
      <Age>25</Age>
      <Sex>M</Sex>
      <Address>13,Microsoft Corp</Address>
      <Salary>$500,000.00</Salary>
    </Guarantor>
  - <Guarantor>
      <Name>Vincent</Name>
      <Age>27</Age>
      <Sex>M</Sex>
      <Address>100,Oracle</Address>
      <Salary>$700,000.00</Salary>
    </Guarantor>
  </Applicant>
```

```
- <Applicant>
    <Name>Dus</Name>
    <Age>23</Age>
    <Sex>M</Sex>
    <Address>12,Intel HQ,Silicon Valley</Address>
    <Salary>$500,000.00</Salary>
    <LoanAmt>$700,000.00</LoanAmt>
  - <Guarantor>
      <Name>Mohan</Name>
      <Age>23</Age>
      <Sex>M</Sex>
      <Address>12,Intel HQ,Silicon Valley</Address>
      <Salary>$500,000.00</Salary>
    </Guarantor>
  </Applicant>
```

The above two diagrams show data represented in the Relational & Hierarchical Model. The Hierarchical Model is Flexible & Semi Structured &hence this Data Model suits our requirements better.

4.2 MAIN OBJECTIVE

The main objective is to integrate FlowMate with TAMINO XML Server. FlowMate consists of several components wherein the data exchange format is essentially in XML. FlowMate currently has ORACLE as its back-end, which is a relational database. Using a relational database to store XML data results in an impedance mismatch as we try to combine two different data models. The relational data model is flat and has a rigid, fixed structure while the XML data model is nested and semi-structured, thus a conversion mechanism has to be adapted to store and retrieve the XML data. The hierarchical nature of XML is lost due to such conversion. Hence there is a need for storing the XML data natively.

The proposed system advocates the use of a native XML repository (TAMINO), to store and retrieve XML documents without any conversion overheads and promises optimal performance.

4.2 SPECIFIC OBJECTIVE

The specific objective is to identify the components of FlowMate and modify them to interact with TAMINO. The project involves modifying one such component of FlowMate called Workflow Definer Tool (WDT). The Workflow Definer Tool is used to define or create workflows. This component is also used to determine the stages that constitute a workflow, the rules that apply to the transition from one stage to another and the user inputs at each stage. WDT, on the definition of a workflow, creates an XML file comprising information pertaining to the workflow. In the current system, this XML file is converted to a format compatible for storage in ORACLE. As proposed, this XML file will now be natively stored in TAMINO through the VB API.

The WDT consists of the WFMAT component that acts as a router for data between WDT and the Database. The WFMAT consists of functions that store and retrieve data from ORACLE. The VB API consists of functions that perform the same tasks with TAMINO and will replace these WFMAT functions.

4.3 PLATFORM SELECTION

Visual Basic

There is several application programming interfaces (APIs) that are offered by TAMINO XML Server. An API for windows platform offered by TAMINO is HTTP Client API for ActiveX. On Windows, this API offers a variety of methods and properties, which enable applications, written in C++, Visual Basic, to access and manipulate documents in a TAMINO database. Since the WFMAT component of WDT is developed in VB, the API to replace the WFMAT component is also developed in VB.

Document Object Model (DOM)

The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

Parsing XML is essential to working with XML documents, but to actually use and manipulate XML content programmatically the Document Object Model (DOM) is used. DOM for XML is an object model that exposes the contents of an XML document. DOM is a standard API that makes it easy for programmers to access elements, delete, add, or edit content and attributes of the XML file. DOM is an object model for representing XML documents in code. It is used to create, modify and access XML documents.

TAMINO being an XML repository relies on DOM for accessing and manipulating XML documents during storage and retrieval. The VB API uses DOM extensively to access and manipulate the XML data in the TAMINO database. The support of DOM is realized in a way that the API's methods supply a DOM object as a result or require a DOM object as input. The standardized DOM methods and interfaces are used to further manipulate the result or navigate through a DOM tree.

TAMINO

TAMINO is a native XML Repository that is used to store the XML data. TAMINO offers a suite of tools to define schemas, insert instances and query the same. The existing relational tables of WDT are restructured and converted to schemas and defined in TAMINO. The outputs of WDT in the form of XML files that adhere to the schemas defined are stored in TAMINO through the VB API.

SYSTEM ANALYSIS & DESIGN – HIGH LEVEL

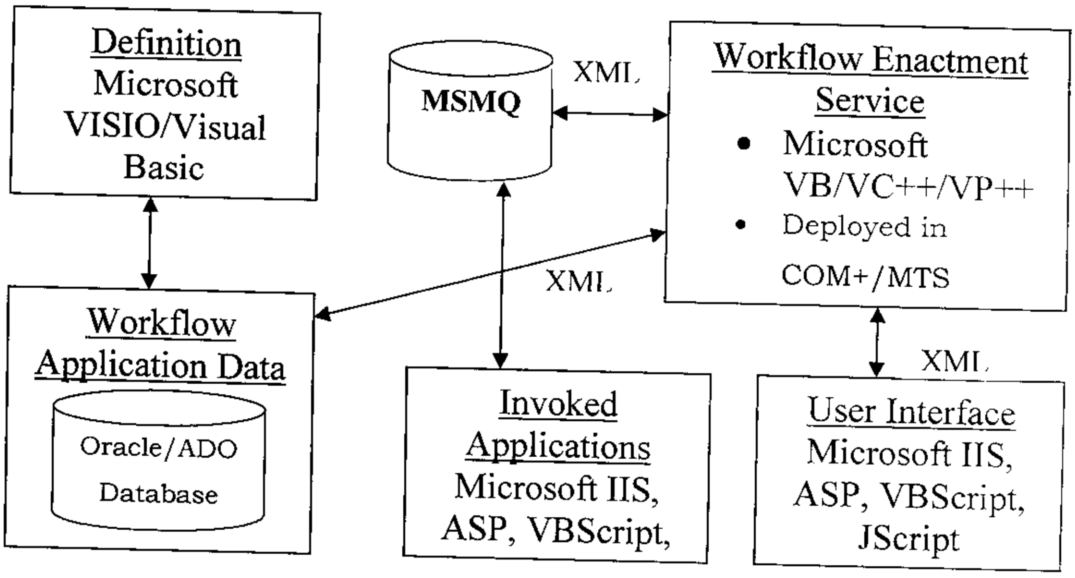
5.1 FlowMate with Oracle 8i

5.1.1 FlowMate Architecture & Description

The FlowMate architecture & description is given in detail in **Section 3.1**

5.1.2 Development Approach

The technology and the tools used is shown in the following diagram


Definition
Microsoft VISIO/Visual Basic
MSMQ
XML
Workflow Enactment Service
• Microsoft VB/VC++/VP++
• Deployed in COM+/MTS
XML
Workflow Application Data
Oracle/ADO Database
Invoked Applications
Microsoft IIS, ASP, VBScript,
XML
User Interface
Microsoft IIS, ASP, VBScript, JScript

5.1.3 FlowMate Components

5.1.3.1 Definition Tool.

The process definition tool consists of two applications - Organization definition tool and the Workflow definition tool. The Organization Definition Tool is a SDI application developed using Visual Basic and the Far point Spread (for grid control). The Workflow definition tool has been developed using the Microsoft Visio COM interface. Visio has been used to define the workflow because Visio is one of the best and easy-to-use product in the market for defining the workflow. Also it would be easier for a user to use an existing user interface instead of learning a new interface.

5.1.3.2 Workflow Enactment Service

Workflow Enactment Service consists of service that uses the process definition data. interprets it and control the instantiation of processes and sequencing of activities, adding work items to the user work lists and invoking applications as necessary. This is done through the following:

- Workflow Client
- Workflow Engine
- Slot Manager

✓ **Workflow Client**

The Workflow Client consists of two components User Object and the WorkflowClient components. The Userobject is responsible for validating the user login and returning the Workflow client object. The workflow client object maintains the worklist for the current user. These components are developed in Visual Basic as COM dlls.

✓ **Workflow Engine**

The Workflow Engine consists of a set of components, which interacts with the database. does the core functions like adding a work, updating a work, deleting a work, delegating a work. These components are developed in Visual Basic and Visual C++. The components are deployed in the COM+/MTS for handling the transactions.

- ✓ **Slot Manager**

The Slot Manager is the only component in the workflow enactment service that maintains state information. This is developed using the Visual C++ . This is a singleton component. Only one SlotManager will be running in the whole FlowMate application. The SlotManager maintains the details related the user like roles, branches and work related information. It also has the current status of the user.

5.1.3.3 Application (User interface)

This layer is one of the important layers and is a unique feature of the FlowMate product. During the process of the process definition itself the User interface is automatically generated using a minimum number of steps and the UI is ready to use. The generated UI can be consumed in ASP or Visual Basic Applications.

5.1.3.4 Monitoring Application

This monitoring application has a set of screens where the administrator users can monitor the execution of the workflow. This application has been done in Visual Basic.

5.1.3.5 Invoked Applications (Scheduler tool)

This Scheduler Tool has a set of screens where a user can manager the tasks and actions. The user can also view the history of the registered tasks this application has been done using Visual Basic. The scheduler has a multithreaded component to monitor for the Time based tasks, which has been done using Visual C++.

5.1.3.6 Database

The database environment in FlowMate is Oracle server 8.1.7 or above. FlowMate can support MS SQL Server with some minimum changes in the code and the stored procedures.

5.1.3.7 Role of MSMQ

FlowMate potentially can be integrated with any third party application. MSMQ has been used in FlowMate for two purposes

- ✓ Asynchronous processing – Inside the Workflow engine the engine need not wait for certain tasks to complete. So this information can be taken into the MSMQ and the engine can continue with the processing while the Scheduler reads the messages and does the necessary tasks for the Engine.

- ✓ Interoperability – The Scheduler accesses the Engine's data from the MSMQ to integrate with the third party applications.

5.1.3.8 Role of XML

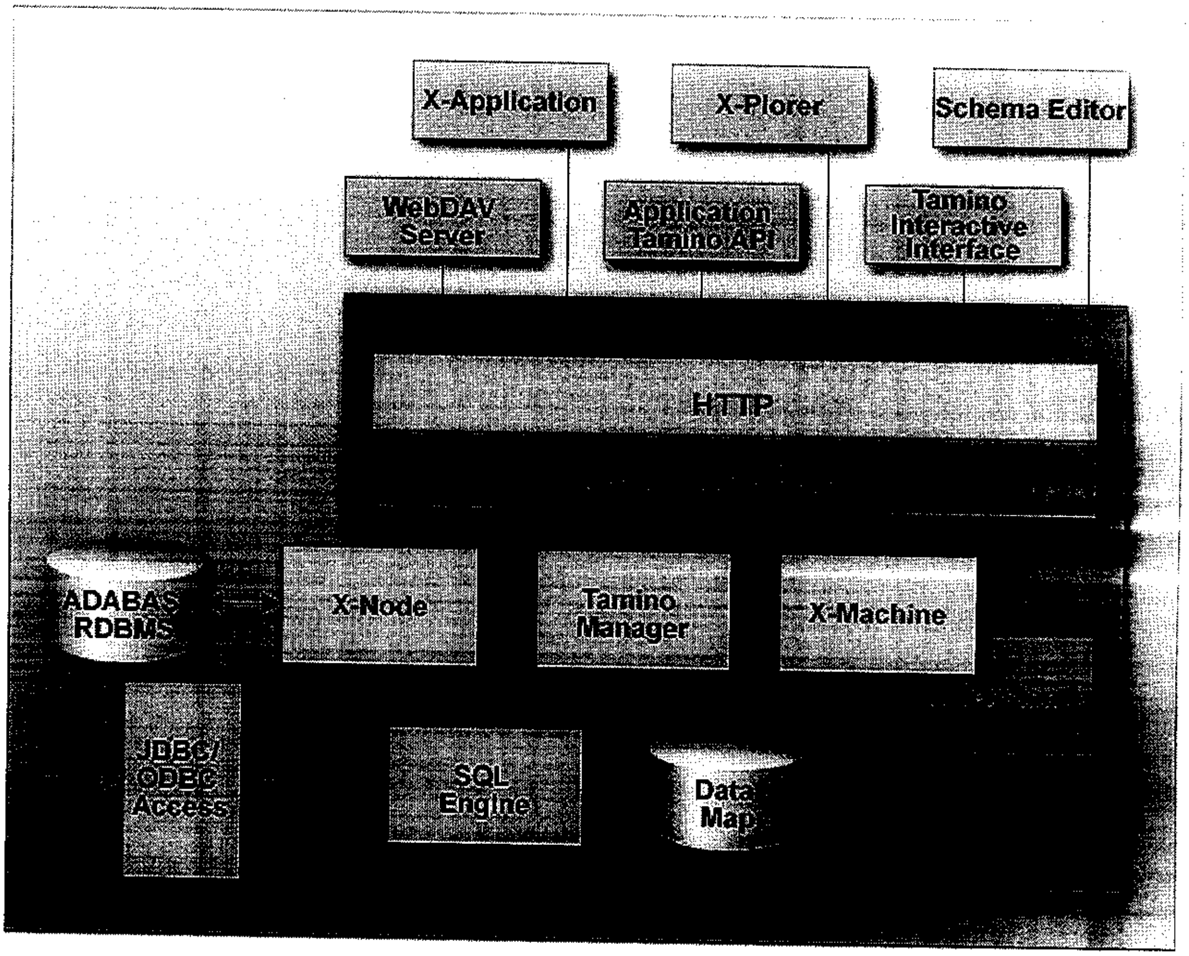
XML is used extensively for the transfer of data across the workflow engine components. Right from definition to execution, XML is used as the medium of communication across various services.

Advantages of this Architecture

- ✓ Allows easy to use and reuse of workflow templates using Microsoft Visio
- ✓ Component based rule processing
- ✓ Ready application service for the defined workflow (Auto generation of user interface, irrespective of the workflow process, stages, attributes on stages)
- ✓ The application can be intranet or web enabled.(IIS and ASP)
- ✓ The application can be scalable and distributable using COM+/MTS
- ✓ Easy integration and interaction with third party application using MSMQ and XML
- ✓ Asynchronous processing using MSMQ
- ✓ Loosely coupled services using MSMQ and XML
- ✓ Easy to deploy using MTS /COM+
- ✓ Better work throughput and lesser processing times as a result of load balancing and caching using MSMQ.

5.2 Tamino Architecture

5.2.1 Tamino Architecture Diagram



5.2.2 Tamino Architecture Description & Deployment

The Tamino XML server is a **Native XML Database (NXD).** The six major components of Tamino are:

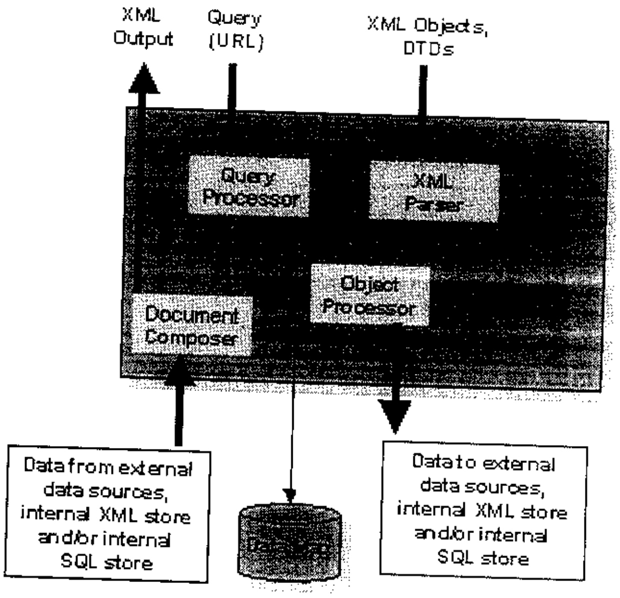
- The X-Machine
- The X-Node
- The Data Map
- The SQL Engine

- The Tamino Manager
- The X-Tension

We will now look into each one of these components in detail.

5.2.2.1 The X-Machine:

The X-Machine is the engine that drives Tamino. Its basic function is to store XML objects natively in and retrieve them from a native-XML data store and various other data sources. It does this based on the schemas defined by the administrator. For storing and retrieving XML objects, the X-Machine is supported by a number of subcomponents, briefly outlined below:


XML Output
Query (URL)
XML Objects, DTDs
Query Processor
XML Parser
Object Processor
Document Composer
Data from external data sources, internal XML store and/or internal SQL store
Data to external data sources, internal XML store and/or internal SQL store

- **XML Parser:**

  XML objects to be stored by the X-Machine are described by their schema stored in Tamino's Data Map. The X-Machine's internal XML parser checks syntactical correctness of the schemas and ensures that incoming XML objects are well formed.

- **Object Processor:**

  The Object Processor is used when storing data using Tamino. XML instances are validated against logical schema. Physical schema defines how the XML instances are stored in Tamino's native XML store, and, for example, SQL data in SQL tables and columns. Support of external data sources is provided by the X-Node.

- **Query Processor:**

  The query language for Tamino is X-Query. The Query Processor optimizes the query along the given schema for resolving X-Query requests.

- **Document Composer:**

  The Document Composer is used when the XML information sets have to be composed. Using the storage and retrieval rules defined in the Data Map, the Document Composer constructs the information objects and returns them as XML documents. The simplest case will be retrieving an object stored internally as XML. In more complex cases, communication with X-Node is required to compose an XML object from non-XML data sources

5.2.2.2 The X node

The X-Node is Tamino's integration component with external data storage systems. The X-Node allows applications to access data regardless of database type or location (for example, in heterogeneous databases). Access to these data stores is provided by ODBC interfaces, which allows access to most relational data stores

5.2.2.3 The Data Map

The Data Map is Tamino's repository. The Data Map contains XML meta-data, XML schemas and its mapping to relational schemas etc., defining the rules according to which XML objects are stored and composed. The Data Map determines how XML objects, embedded in XML documents, will be mapped to physical database structures, whether they reside internally or externally. This way, the Data Map allows existing databases to be enabled for XML technology and the Web. The Data Map therefore contains the information required for the following functions:

- ✓ Validation against logical schema
- ✓ Storage and indexing of XML objects within Tamino
- ✓ Mapping of data to different data structures (for example, relational databases) to enable the integration of existing data
- ✓ Executing user-defined application logic using a Server Extension Function associated with an object.

The definition of the schemas in the Data Map is supported by a graphical tool (the Tamino Schema Editor) that guarantees the creation of error-free XML syntax and provides some default specifications.

5.2.2.4 The SQL Engine

The SQL Engine manages Tamino's internal SQL Store. The SQL Engine executes SQL statements (*DML, DDL, DCL*). The SQL statements accepted by the SQL Engine include SQL 2 Entry Level. Apart from the functionality of this standard, some higher levels of SQL 2 are supported.

5.2.2.5 The Tamino Manager

The Tamino Manager is Tamino's administration tool. The Tamino Manager is an administration tool implemented as a client-server application and is integrated in the System Management Hub, Software AG's multi-platform environment for the unified management of Software AG products. The Tamino Manager provides a GUI that runs in standard web browsers. For basic administration functions, (create database, start/stop server, backup, restore, load. etc.). the Tamino Manager uses the System Management Hub installed on the node where the Tamino server to be administered runs.

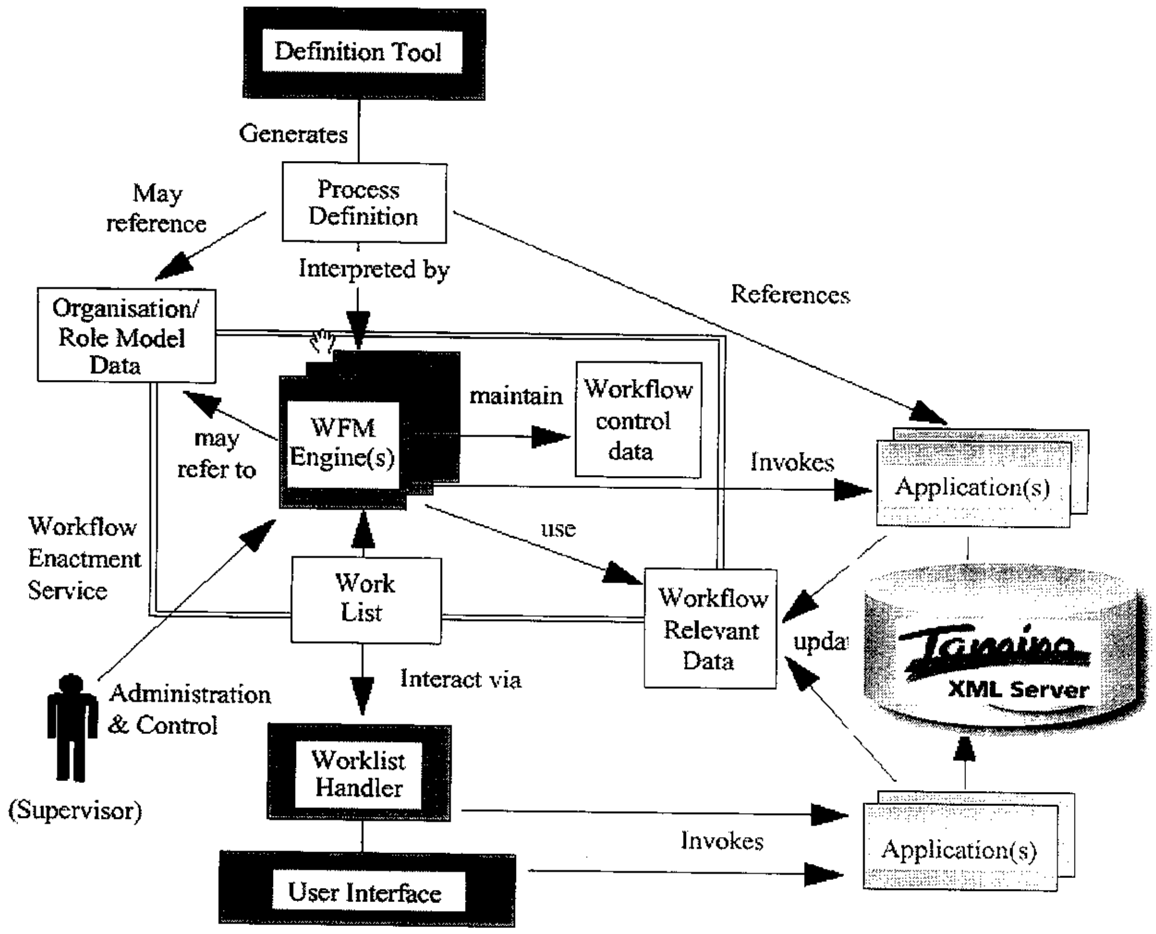
5.2.2.6 X-Tension

Tamino's X-Tension component allows calls to user-defined functions, so-called Server Extensions. Based on schema definitions stored in the Data Map, an XML object can be "mapped" to a user-defined function which is then executed, either on parsing an incoming object or when composing a retrieved object.

Server Extensions are installed as functions of the Tamino Server. A typical user-defined function is one that handles data in some specific way that cannot be anticipated by a "standard" function provided by Tamino. Such extension functions are installed in the Tamino Server using the Tamino Manager. Once plugged in, these extensions are not distinguishable to the end-user from Tamino's "standard" functions.

5.3 FlowMate with Tamino XML Server

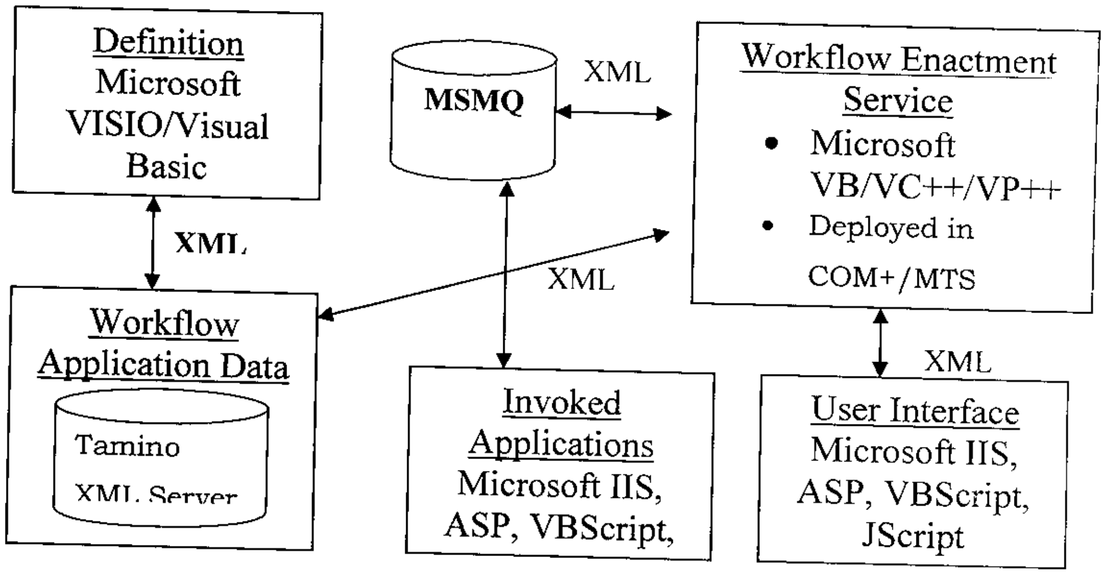
5.3.1 Architecture



The architecture diagram shows clearly what the entire project is about. On comparison with the FlowMate architecture, one finds that the Oracle 8i database is replaced by the Native XML Server Tamino 3.1.2.1. The architecture description is the same as that of FlowMate in all respects other than the Database. The Database is accessed by the WDT & ODT by means of the DLL that we write.

5.3.2 Development Approach

The technology and the tools used in the architecture is shown in the following diagram


Definition
Microsoft VISIO/Visual Basic
MSMQ
XML
Workflow Enactment Service
• Microsoft VB/VC++/VP++
• Deployed in COM+/MTS
XML
XML
Workflow Application Data
Tamino XML Server
Invoked Applications
Microsoft IIS, ASP, VBScript,
XML
User Interface
Microsoft IIS, ASP, VBScript, JScript

Communication between **all** the modules is through XML. Previously the Definition Component transferred data to the Database component by means of recordsets.

5.3.3 Components

FlowMate's integration with Tamino can be considered as a typical migration exercise; architecturally it is similar to the existing implementation. The data access component is modified to handle the Tamino integration.

The Components of FlowMate with Oracle are the same as FlowMate with Tamino except for the Database component & how FlowMate communicates with Oracle. Please refer to Section 5.1.3 for reference to the FlowMate Components. For a detailed report on Tamino XML Server, refer to section 5.2.

5.3.4 Advantages of this Architecture

This architecture combines the best of both – that of FlowMate & that of Tamino. The advantages of the FlowMate architecture are specified in Section 5.1.5. That of FlowMate integrated with Tamino is specified here

Data Management

Tamino is a multi-threaded server that benefits from Software AG's proven database technologies such as indexing, compression, caching, as well as high data volume capabilities. The result is an XML-based data management system capable of handling large volumes of data (high data throughput) and managing concurrent user requests very efficiently (high performance).

Scalability

Tamino achieves a high degree of scalability by virtue of XML's ability to encompass additional types of information items without making existing information invalid.

Security

Apart from providing an authorization mechanism to control access to XML objects stored in the XML Store, Tamino integrates security concepts at different levels (for example, the transport and application level).

External Database Integration

Possible to connect to External Databases like Mainframe's, RDBM's etc. through the use of X-**Node.**

Native XML Database

Tamino is a Native XML Database and hence no conversion procedures are required to store XML files & it is better than a XML Enabled Database like Oracle 9i that ultimately stores XML in Flat Structure.

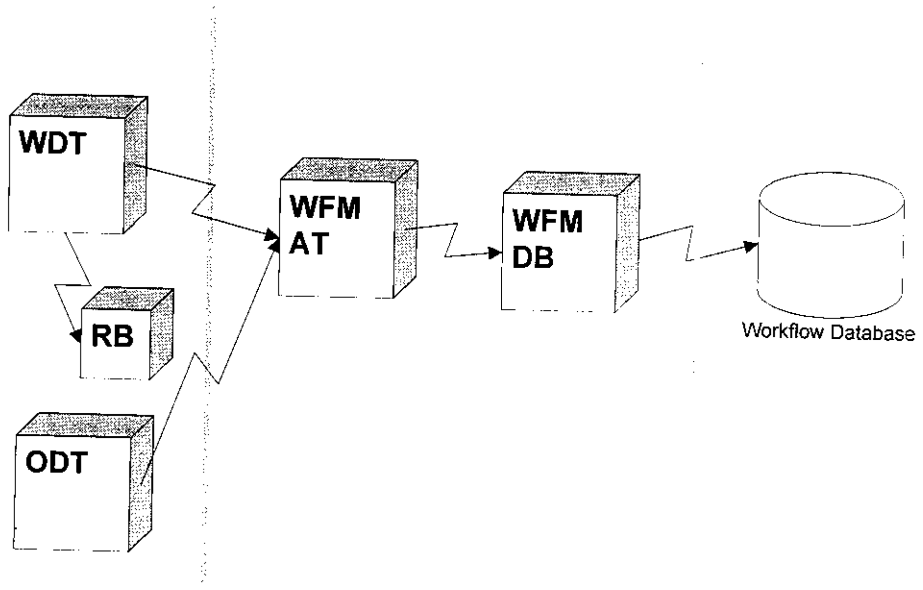
- With the use of the Data Map it is possible to store data in the XML or SQL store
- The vast amounts of processing power required to deal with the Table structure are done away since a Native XML Database is used.
- Querying is faster.

SYSTEM ANALYSIS & DESIGN – LOW LEVEL

6.1 Component Design

The component design gives a brief description of components of Workflow Definition Tool.

6.1.1 Architecture Deployment:



In this Tool, the following modules and components are:

a. **ODT →** Organization Definer Tool exists as an Application, from which the Organization and user related data is communicated to the Database through the WFMAT component.

b. **WDT →** Workflow Definer Tool exists as an Application, from which the Workflow definition related data is communicated to the Database through WFMAT component.

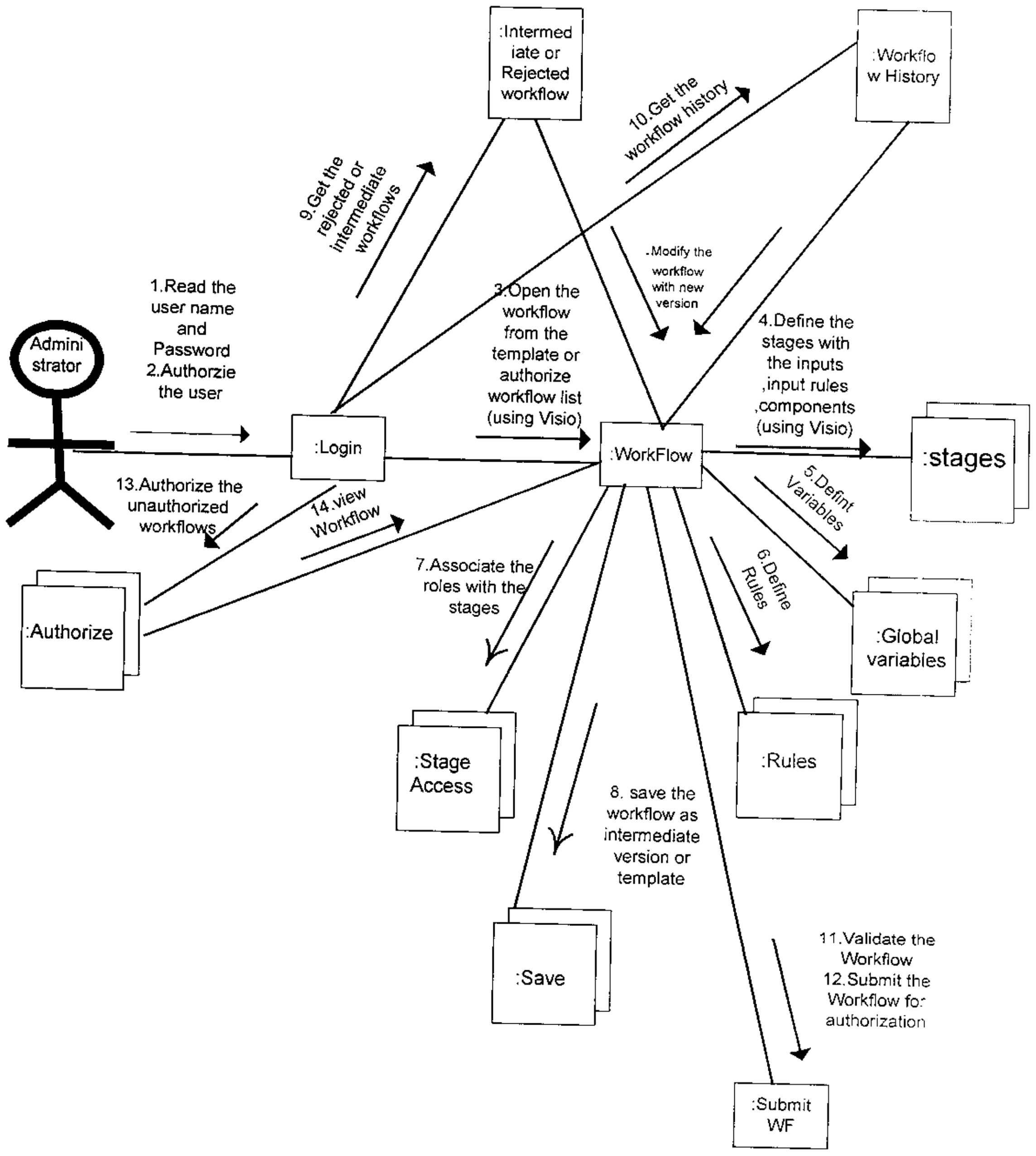
c. **RB →** Rule Builder is for parsing the rules defined for every stage. This component is called by WDT tool for validating the Rule as well as getting the Rule XML from the rule Expression.

d. **WFMAT →** Workflow AT component acts as a router for data between Definition Tools and the Database.

e. **WFMDB →** All the database related functionality for WDT and ODT is taken care by this component. This is called from WFMAT component.

6.1.2 Collaboration Diagram

The Collaboration diagram describes how the WDT collaborates with the various schemas associated with it.



Administrator
1.Read the user name and Password
2.Authorzie the user
:Login
9.Get the rejected or intermediate workflows
:Intermediate or Rejected workflow
3.Open the workflow from the template or authorize workflow list (using Visio)
10.Get the workflow history
:Workflow History
.Modify the workflow with new version
:WorkFlow
4.Define the stages with the inputs ,input rules ,components (using Visio)
:stages
5.Defint Variables
:Global variables
6.Define Rules
:Rules
13.Authorize the unauthorized workflows
14.view Workflow
:Authorize
7.Associate the roles with the stages
:Stage Access
8. save the workflow as intermediate version or template
:Save
11.Validate the Workflow
12.Submit the Workflow for authorization
:Submit WF

Login:

Get the Task rights for the current login user.

New Workflow:

Allowing the user to define his new workflow, stages and rules. Workflow and Stages are defined using Visio. The drawing of stages and the rule connecting are interpreted and then stored in the database through WFMDB component interface. Along with this the Visio XML for the controls (workflow and stages) is also saved.

Define Stages:

Allow the user to define the Stages from stage templates or new stage. This is also done using Visio. If it is an Input stage then get the input rule from the user. If it is an Auto stage then get the component details from the user. If it is a different workflow stage then get the workflow name from the user. Allow the user to save the stage as stage template.

Define variables:

Define the Global variables for the workflow.

Define Rules:

Allow the user to define the rule expression from the rule templates or a new rule expression using the stage inputs and the global variables, which is already defined. Accept the component-based rules also.

Allow the user to save the rule as rule template.

Stage Access:

Associate the Roles with the stages.

Open Workflow:

Allow the user to open and modify the existing authorized version of the workflow. Existing workflow diagram is displayed in Visio. Allow the user to modify the stages, rules and stage access. Allow to define the new stages or rules from the templates or new.

Workflow History:

Get all the authorized and all the versions of the workflows. Allow the user to open the workflow from the workflow history, allow the user to modify the stages, rules and stage access. Allow to define the new stages or rules from the templates or new.

Intermediate Or Rejected Workflows:

Get all the intermediate or rejected workflows for the current user. Allow the user to open the workflow from the workflows. Allow the user to modify the stages, rules and stage access. Allow to define the new stages or rules from the templates or new.

Save:

Save the modified workflow as in the intermediate stage or as a workflow template.

Submit:

Submit the workflow for the authorization.

Authorize Workflow:

Get the All the workflows, which are submitted for the authorization except the current user's workflow. Authorize the workflow and set the workflow status as Accepted or rejected. If the workflow is accepted then set the new version for the workflow.

6.2 Feasibility Analysis:

The new system proposed should emulate the existing system and enhances it by breaking free from the former disadvantages and should ultimately improve the productivity of the system. With this as the objective, the feasibility, in its various dimensions is discussed below.

6.2.1 Technical Feasibility

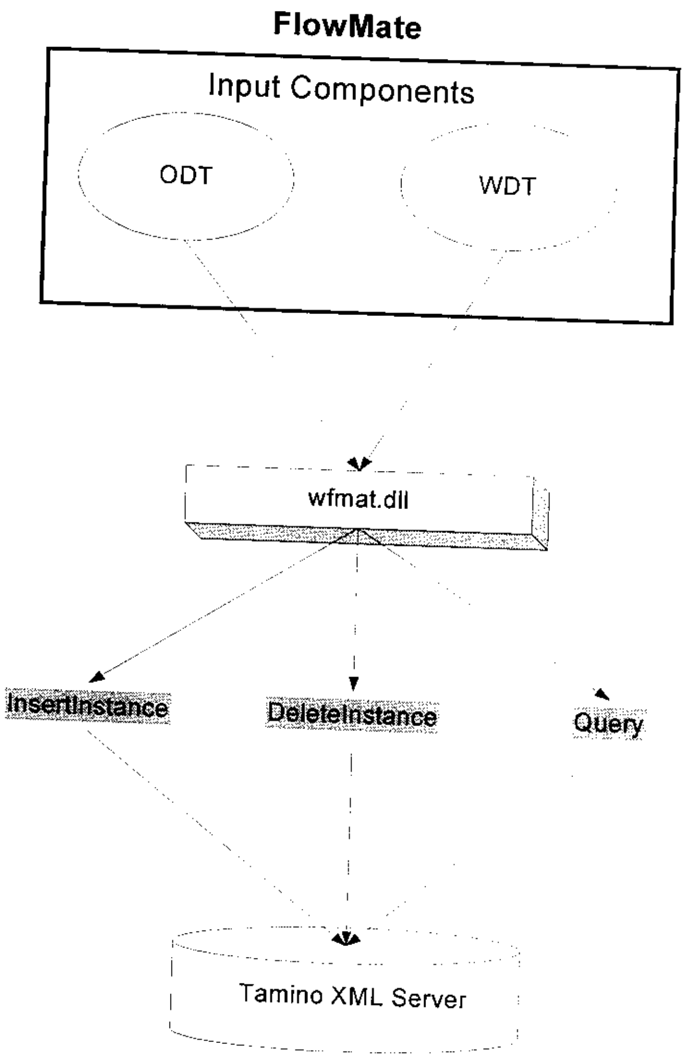
The incapability of fitting a **semi structured** XML file into the Oracle without conversion procedures, pointed all fingers to Tamino as the perfect solution. Tamino a Native XML Repository is capable of storing & retrieving XML files natively as is the requirement in FlowMate. Moreover the Tamino comes packaged with efficient tools with which defining schemas, inserting XML instances, querying & defining the Data Map are possible.

6.2.2 Operational Feasibility:

As the system becomes operational the end users might not even notice the difference unless they notice it while saving & retrieving the WorkFlow & while running the client application. **Query Access** would be faster since the content of 23 tables are now dealt with in just **three schemas.**

6.3Code Design:

The Coding had to be changed for the Data Access layer for the WDT component. The WDT calls the **wfmat.dll**, by which it accesses the Tamino Server to add, delete & modify XML Files.



Function call DFD

The code in the **wfmat.dll** has got functions for insertion, updating, deletion & fetching of elements values pertaining to each of the three WDT Schemas. A few functions are as follows:

InsertInstance (SchemaName as String, Filepath as String):

This purpose of this function is to insert an instance of the WorkFlow Schema. The parameter schema name is to identify which schema to add the instance to (WorkFlow, StageInputs or Rule). The parameter FilePath is knows where the temporary XML file is stored. This Function does not return any value.

InsertInstance (SchemaName as String, DOMObject DOMDocument):

This function also does the function of insertion. The only difference between the above function being it taking a DOM Document as a parameter. Depending on the need either the previous Insert function or this one is called.
This Function does not return any value.

DoQuery (QryString as String)

This function performs the specific query passed to it as a String. If the query is valid this functions returns all the instances pertaining to that query to the Object that called this function.

ModifyInstance(SchemaName as String, NewNode as Object, ExistingNode as Object)

The New Node Contains the Element with its Attributes & Text, which will replace the existing node. The differences between these two nodes may be minor like an attribute being set to true or False, depending on the context. In some cases the entire element values & attributes may have to be changes.

DeleteInstance (SchemaName as String, DelQry as String)

This function deletes all instances pertaining to the query if the query is valid.

6.4 Database Design:

The WDT uses 3 schemas – WorkFlow, StageInputs & Rules that are shown below in figure 5.2. These 3 schemas were derived from the existing Database tables, which were used in the previous version of FlowMate with Oracle. The tables from which the schemas were extracted are listed below:

- WFMWorkFlow
- WFMStage
- WFMStageInputs
- WFMRule
- WFMRuleStage
- WFMRuleStageOutput
- WFMRuleStageOutputMultiple
- WFMListValues
- WFMStagedDocuments
- WFMGlobalVariables
- WFMDeadLine
- WFMConfig
- WFMGroupOrder
- WFMStyleSheets
- WFM2Dstrucutre
- WFM2Ddetails
- WFMStage2Dstructure
- WFMActionType
- WFMStageActions
- WFMStageActionOutput
- WFMStageActionInputs
- WFMStage2Ddetails

Figure 6.4.1 – Schema for WorkFlow

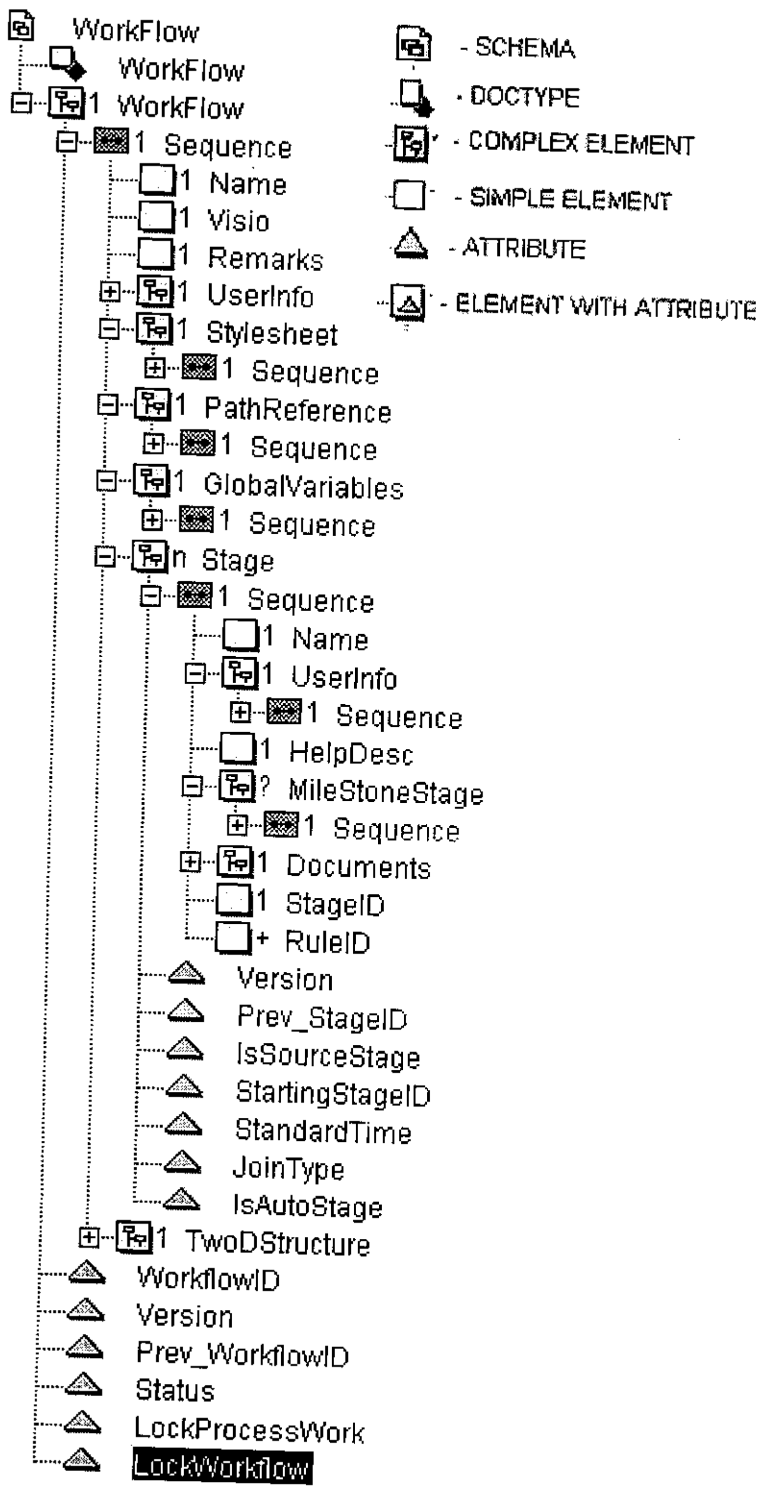


Figure 6.4.2 – Schema for StageInputs

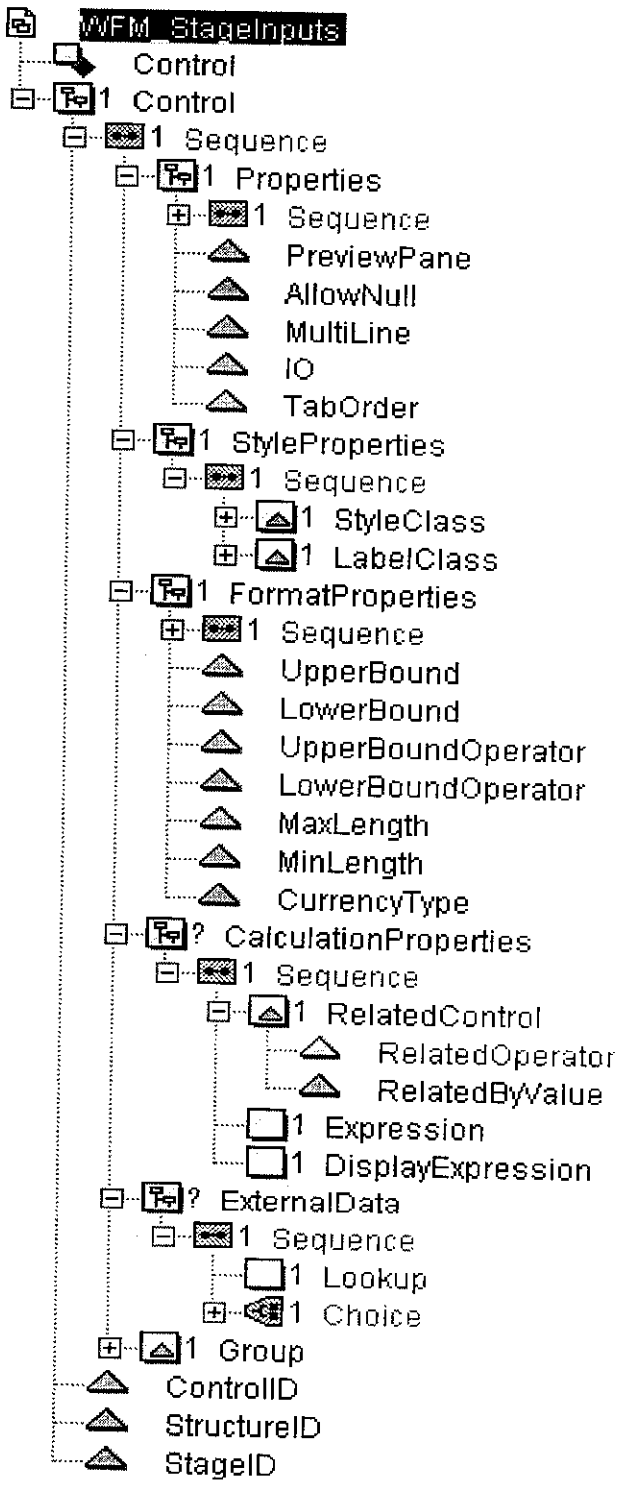
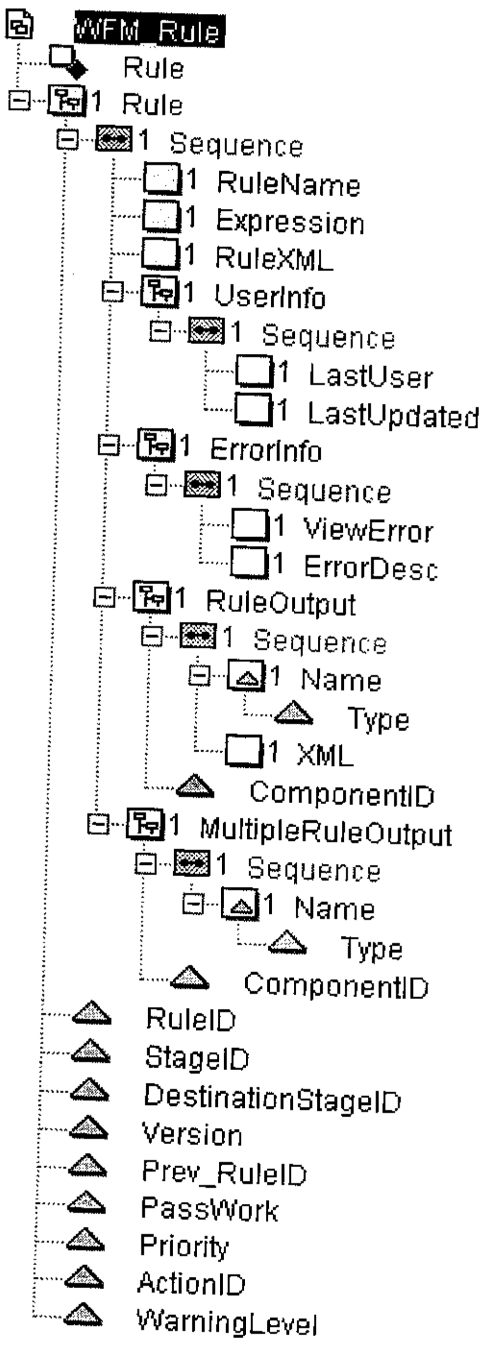


Figure 6.4.3 – Schema for Rules



The schemas were designed using the **Asset Oriented Model (AOM)** as a guideline. This design breaks free from the drawbacks faced when designing a schema using the Entity Relation Model. The WorkFlow Schema is the Business Object (AOM term), which in traditional RDBMS Concepts can be equated to main table containing the foreign keys. The following elements of the WorkFlow Schemas are joined:

- StageID Element is joined to the StageID attribute of the StageInput schema
- RuleID Element is joined to the RuleID attribute of the Rule Schema
- ControlID Element is joined to ControlID attribute of the StageInput Schema.

The Schema's are in the TSD 3(Tamino Schema Definition) format since it was designed using the Schema Editor tool, which comes with Tamino. The TSD 3 specification is a Subset of W3C's XSD (XML Schema Definition). This Schema structure is only valid for use in Tamino.

6.5 Validation Checks:

In the classical relational database, integrity rules and triggers are the means to maintain the integrity of the information stored in the database. Integrity means that the constraints defined in the conceptual model are not violated and that the data structures defined in the conceptual model are kept intact. This is possible by applying integrity rules and triggers within the same transactional context as the operations that modify the stored information.

The consequence is that in the general case the resource manager (i.e. the database) is the wrong instance for the enforcement of data integrity. In many cases this task is better left to the application logic, or to appropriate middleware.

In the following sections we show how constraints can be defined for XML documents. The technique used here is based on XSLT. This is probably not the most efficient way to implement integrity rules, but serves well for rapid prototyping purposes. Typically, this technique is applied by an application that wants to write to a database. By using it within a Tamino session context (with isolation level "_shared" or "_protected"), we can guarantee transactional safety

6.5.1 Simple Constraints

Constraints are used to add more meaning to a model. During the definition of the XML schemata we have already added a considerable set of constraints to our model data types. Each data type such as string, float or integer constrains the value domain of an element or attribute.

Additional constraints are enumerations or length specifications such as total Digits and max Length.

Another type of constraints is the cardinality constraint, which can be defined in schemata using minOccur and maxOccur. For example, by decorating the element

```
<xs:element name = "Guarantor" type = "xs:string"
        minOccurs = "2" maxOccurs = "unbounded" >
```

In a Loan Application WorkFlow, we set up a constraint that a Person must provide details of two Guarantors for his loan to be processed.

6.5.2 Cross Field Constraints

To understand these types of constraints better we shall illustrate an example. Visualize a Loan Application WorkFlow, which requires person to provide details of two Guarantors, but if the person provides the details of a bank manager as the first guarantor, he need provide a second guarantor.

To make it short: there is no way in XML Schema to define this sort of constraint, so XML Schema aware validating parsers will not check these constraints. Extra software is required for that task. A popular tool for all constraint processing that goes beyond data types and cardinalities is *Schematron* (http://www.ascc.net/xml/schematron/). Schematron can be used independently or in conjunction with DTDs and XML Schema.

Schematron allows the definition of assertions for a given document type, similar as one would define integrity rules for SQL tables. However, the syntax used to formulate these assertions is XPath. A rule for our person-guarantor constraint could look as follows:

```
<rule context="Guarantor">
  <assert test="@profession='Bank Manager' >
            <set minoccurs(Guarantor=1)>
            Only one Guarantor is required
  </assert>
<assert test="@profession='Others' >
            <set minoccurs(Guarantor=2)>
         Two Guarantors are required.
  </assert>
</rule>
```

The rule first specifies a context node (via an XPath expression) on which the following assert clause is applied. The assert clause defines a test condition (again an XPath expression) that must hold. If the test fails, the content of the assert clause (i.e. the message) is written to the output stream. In this case we have used plain text, but we could as well have used some XML expression to generate a machine-readable error report.

How does this work? All constraints for a given document type are defined within an Schematron schema file. This file is then compiled by the Schematron compiler (the compiler is actually an XSLT style sheet). The result of the compilation process is another XSLT style sheet, which can then be applied to our document instances. The output of this last step is the final error report.

Applying the rule given above to the following document instance:

```
<?xml version="1.0"?>
<PersonalReferences>
<Guarantor>
Project Leader
<Guarantor>
<PersonalReferences>
```

Leads consequently to the error report:

Two guarantors are required.

We can adopt the XPath notation used in Schematron to include constraints into our conceptual model.

Testing And Implementation

7.1 Types of Testing Done

Testing, Debugging & Quality Assurance

Testing:

Testing is the process of examining something with the intention of finding errors. While testing may reveal a symptom of an error, it may not uncover the exact cause of the error.

Debugging:

Debugging is the process of locating the exact cause of an error, and removing that cause. Software quality assurance assures the effectiveness of a software quality program within a software engineering organization.

Testing activities can begin and proceed in parallel with concept definition, Analysis, Design and programming. When testing is correctly interleaved with development, it adds considerable value to the entire development process.

While testing software the following things are to be taken into consideration.

- UI
- Functionality
- Usability
- Reliability

Every application is unique in its own way. The testing methodology used for one application may not be the same for another. But there exist a lot of methodologies, which help a tester to choose one, or a combination of these for his/her application needs.

With this as the objective to achieve the following testing strategies were adopted to ensure that the system is bug free and thus to meet the vision High Quality, Reliable system.

- Unit Testing
- Integration Testing
- Validation Testing
- System Testing

Unit Testing

Initially, tests were focussed on each module individually, assuring that it functions properly as a unit. In all the unit testing done, heavy use of white-box testing techniques were employed, exercising specific paths in a module's control structure to ensure complete coverage and maximum error detection.

As Kaner, Falk and Ngyen clearly points out, the test cases were selected for all the sub-units, so that they meet the following criteria:

- A good test case is one that has a high probability of finding an as-yet undiscovered error.
- Not redundant and "Best of the breed".

The way in which the White-Box test strategy was employed was to ensure that the test cases could

- ✓ Guarantee that all independent paths within a module have been exercised at least once
- ✓ Exercise all logical decisions on their true or false sides
- ✓ Execute all loops at their boundaries and within their operational bounds and
- ✓ Exercise internal data structures to assure their validity

As Myers suggests that when a module performs external I/O, additional interface tests must be conducted, the following checks were made:

- Checking of attributes for their correctness
- Proper functionality of OPEN/CLOSE statements
- Handling of end of file condition, I/O errors, buffer problems and textual errors in Output information.

Even though, the above listed testing strategies classically hold good for current applications they fail to capture the newer and newer bugs that the development scenario churns up from time to time. This implies the indispensable need for the use of additional techniques to capture and eliminate the seemingly endless number of all the new glitches.

At the unit level testing, **Link Testing** was carried out. **Link Testing** principally involves testing of links of an HTML or a CFM page. This was done using automated HTML validates as obviously a manual process gets impossible for a huge site.

7.2.User Interface Testing

User interfaces were to be tested specifically to identify loopholes, if any, in the interface provided.

- The Form where the inputs are got from the user are tested if they adhere to their data types. These gave prompt messages that the user could understand to input values.

Integration Testing

Incremental integration test strategy was employed which grows by constructing small segments to the entire system. The so-called Black-box testing methodology was frequently adopted to test the integrated system and this was performed in relatively later stages of testing process.

Since integration testing has to check the functionality of the system as a whole, the exact sequence of steps involved in construction of a WorkFlow, Authorizing, creation of client application are verified.

Validation Testing

Validation testing was then performed to check whether the designed system met the necessary requirements specified in the early requirements analysis phase and the system was found to be matching with the requirements.

System Testing

It is actually a series of different tasks whose primary purpose was to fully exercise the computer-based system to verify that all system elements have been properly integrated and perform allocated functions. On the whole the system was tested for the various paths it can take right from the submitting of the report to the Approval or Rejection of the WorkFlow. Thus the system has been found to be wholesome in it, completing the development of the application.

Acceptance Testing

The fact that the Client has the last word is true for any business enterprise and the IT is no exception to it. The developed system was deployed at the client side and the initial response about the system acceptance was obtained. Things are left to the test of time. Only time can ferret out those deeply buried glitches. Let us hope that time will flunk at least this once!

7.3 IMPLEMENTATION

7.3.1 Hardware Interfaces for WFMS

Pentium III 800MHz with 128 MB RAM and 10GB Hard Disk

7.3.2 Software Interfaces

The following are the software requirements for the system

- Microsoft Visual Basic 6.0
- DOM
- Tamino XML Server
- XML 4.0
- Microsoft Visio 2000
- Windows 95/98/NT

7.3.3 Communication Interfaces

User Communicates with the system Dom interface. Data transfer is done by using XML.

Scope for Further Development

This Development phase has just migrated the WDT component of FlowMate to Tamino. The WDT component uses Tamino as its backend while all other components of FlowMate use Oracle as before. Since the migration of WDT has proven worthwhile in terms of data access in all aspects, it is suggested that all components of FlowMate be migrated to Tamino.

Provisions have been made for including additional features and controls to the existing system. Basically Software restructuring is carried out. Software restructuring modifies source code in an effort to make it amenable to future changes. In general, restructuring does not modify the overall program architecture. It tends to focus on the design details of individual modules and on local data structures defined within modules.

Suitable and meaningful comments have been added to the codes of each module wherever required. This has been done so as to enable future enhancement and upgradation of the software. When some other programmer has to add new feature, he can understand each part of the code by its comments. The comments have been added to both the front-end and the back-end coding.

Conclusion

The project entitled FlowMate with Tamino XML Server has been successfully completed and installed. The migration to Tamino has been a real Eye Opener due to the fact that the real powers of an XML repository far exceeded expectations. By using Tamino as the backend for FlowMate the problems like Round Tripping loss of data and Impedance Mismatch have been eliminated, thus increasing the efficiency and effectiveness of using FlowMate in terms of lesser processing time and easier access to Data.

Bibliography

Books

1. Introduction to XML, its major applications, and tools by Charles F. Goldfarb's XML Handbook™ Fourth Edition

2. Designing XML Internet Applications by Michael Leventhal, David Lewis, Matthew Fuchs

3. Designing XML Databases by Mark Graves

4. XML and Web Services by Ron Schmelzer Et AL, Pearson Education

Websites

www.softwareag.com\tamino

www.w3schools.com

www.aomodeling.org

www.xml.com

www.w3.org/XML

www.msdn.microsoft.com/xml/

Data Dictionary

Administrative Tool – The Administrative Tool is used for populating the WorkFlow Data Sources. It gives an interface UI for the user (System Administrator) to define the Workflow, stages and rules.

Architecture- The logical and physical structure of a system, forged by all strategic and tactical design decisions applied during development.

Client Information System- Client Information System is the module for interacting with the client component when different operations are performed on works.

Method- An operation upon an object, defined as a part of the declarations of a class.

Object oriented Analysis – A method of analysis in which requirements are examined from the perspective of the classes and objects found in the vocabulary of the problem domain.

Object Oriented Design – A method of design encompassing the process of Object-Oriented Decomposition and a notation for depicting both the logical and physical models of the system under design; specifically this notation includes class diagrams, object diagrams, module diagrams and process diagrams.

Parser – The Parser checks for the syntax of the rule expressions and validates the rule expression from the rule builder.

Process – The activation of a single thread of control.

Rules – This involves defining a set of conditions based on which a work moves from one stage to another. Rules are categorized according to the stages of work. Each Rule is applicable for a particular stage.

Rule Builder – A Rule Builder will generate the XML file. The Rule Builder frames the rules according to the requirements of the systems administrator for the organization.

Rule Executor – It is the rule processing module for deciding the destination stages of a work. This is part of the Rule Manager and will not be exposed as a component.

Rule Manager – A Rule Manager is the controlling module that exposes interfaces for the rule processing and for the rule database maintenance.

Stages – This involves defining the various stages that constitute the path. The entire work will be done through various steps, called the stages of the work.

Tree Builder – The Tree Builder builds the tree structure and converts the rule expression into a interpretable form in XML.

Values – Values are the inputs to a stage and corresponding resultant of the stage. Each stage has some information pertaining to the work and requires input from the user.

WorkFlow – This involves designing the path that should be taken to complete a specified work.

WorkFlow Definition Tool – The WorkFlow Definition Tool is used to capture the WorkFlow in the XML format.

WorkFlow Connector – The Workflow Connector is the exposed interface for the client application.

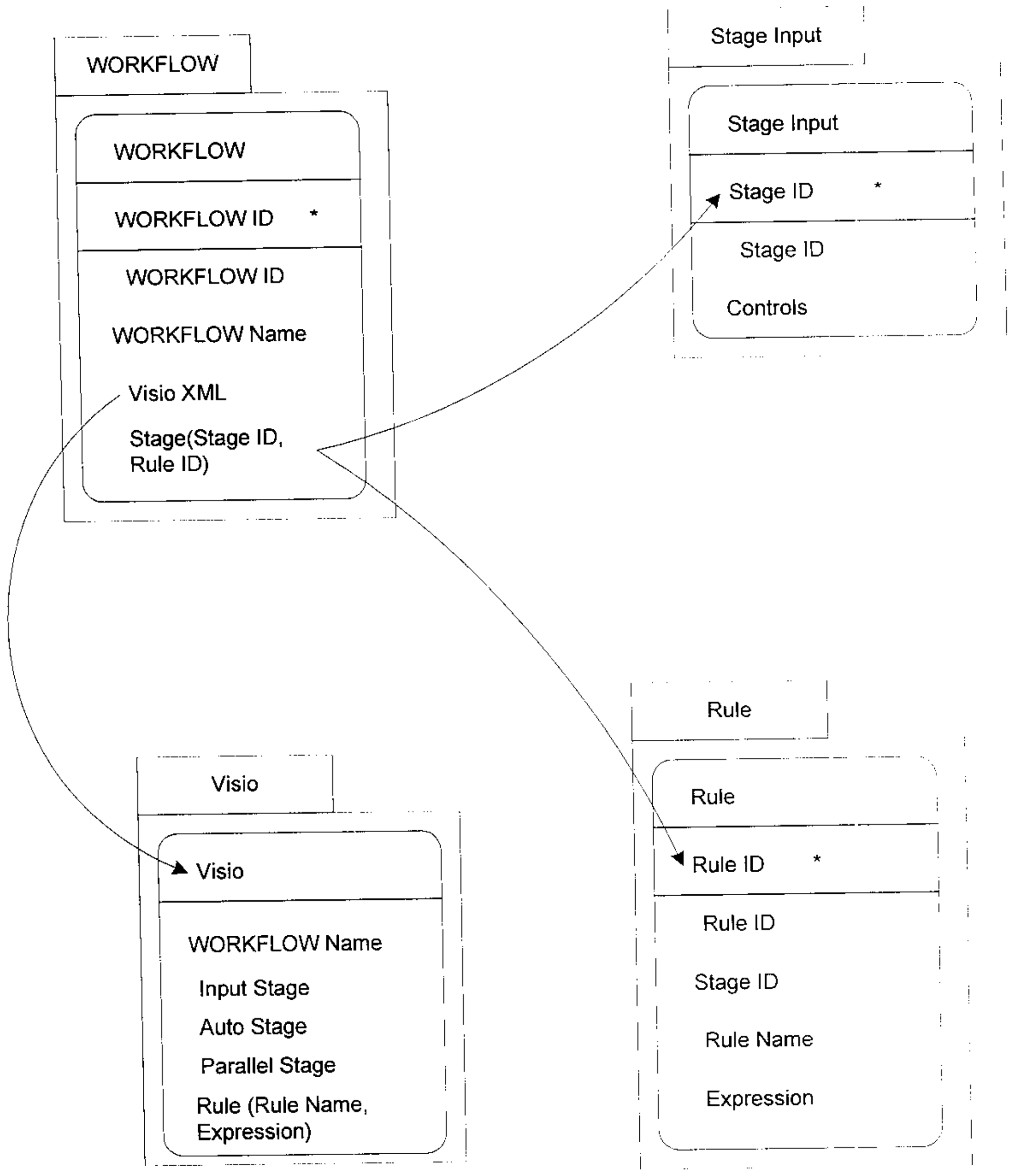
WorkFlow Server – The WorkFlow Server does all the data transmission between the components. It uses different XML files for data transmission among components. It is the connecting point for the WorkFlow Connector to the other components.

WorkFlow Management System – The WorkFlow Management Systems is a generalized model for managing WorkFlow and customizing it as per the client requirements. This involves developing a general application API and a server API which will interact with the database.

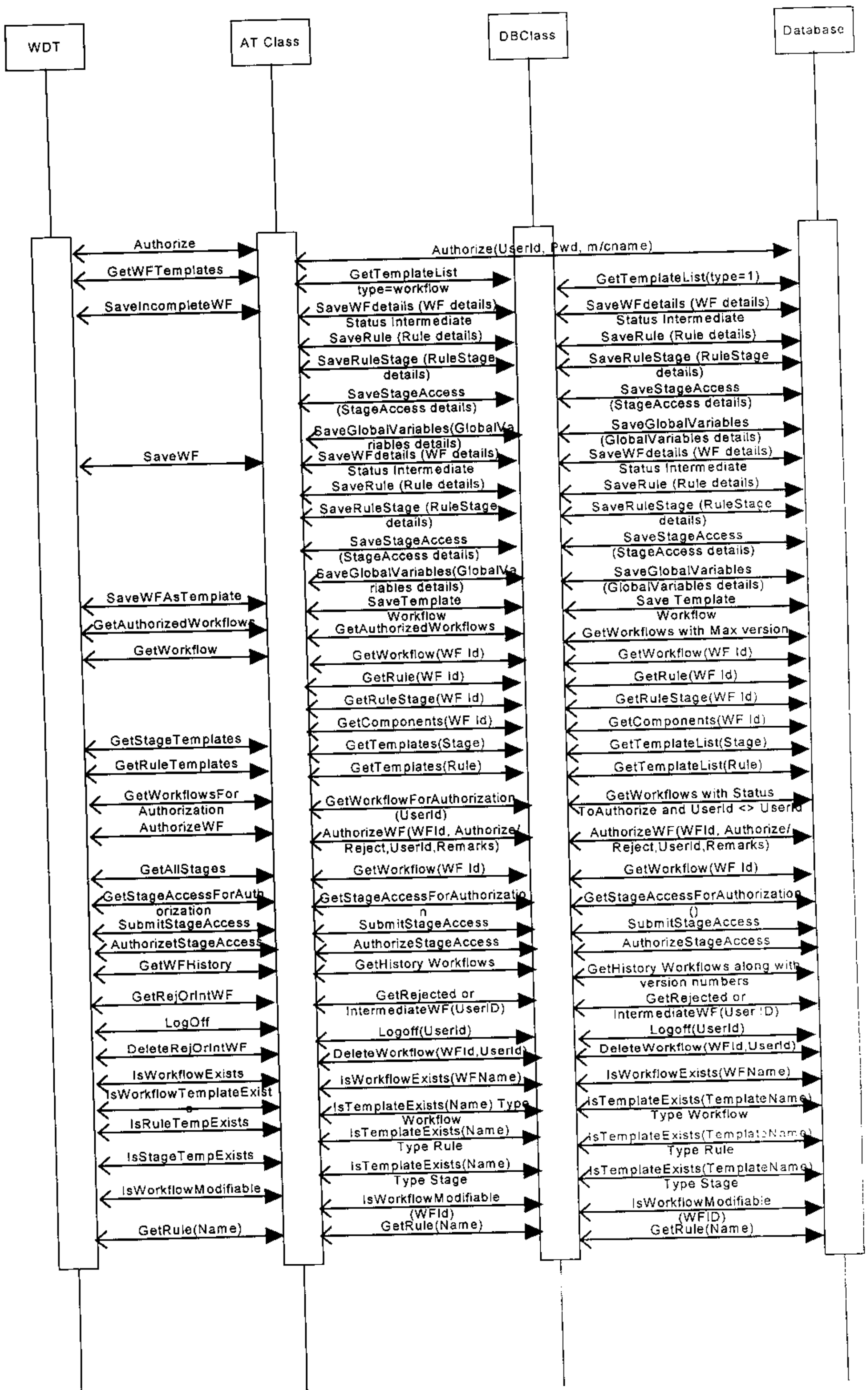
WorkFlow Information Manager – The WorkFlow Information Manager is a wrapper over the data sources specific to the WorkFlow Engine/System. This is part of the Work Manager and will not be exposed as a component to the external world.

Work Manager – Work Manager is for managing works in the WorkFlow system.

AOM For Schemas



SEQUENCE DIAGRAM


WDT
AT Class
DBClass
Database
Authorize
Authorize(UserId, Pwd, m/cname)
GetWFTemplates
GetTemplateList type=workflow
GetTemplateList(type=1)
SaveIncompleteWF
SaveWFdetails (WF details) Status Intermediate
SaveWFdetails (WF details) Status Intermediate
SaveRule (Rule details)
SaveRule (Rule details)
SaveRuleStage (RuleStage details)
SaveRuleStage (RuleStage details)
SaveStageAccess (StageAccess details)
SaveStageAccess (StageAccess details)
SaveGlobalVariables(GlobalVariables details)
SaveGlobalVariables (GlobalVariables details)
SaveWF
SaveWFdetails (WF details) Status Intermediate
SaveWFdetails (WF details) Status Intermediate
SaveRule (Rule details)
SaveRule (Rule details)
SaveRuleStage (RuleStage details)
SaveRuleStage (RuleStage details)
SaveStageAccess (StageAccess details)
SaveStageAccess (StageAccess details)
SaveGlobalVariables(GlobalVariables details)
SaveGlobalVariables (GlobalVariables details)
SaveWFAsTemplate
SaveTemplate Workflow
Save Template Workflow
GetAuthorizedWorkflows
GetAuthorizedWorkflows
GetWorkflows with Max version
GetWorkflow
GetWorkflow(WF Id)
GetWorkflow(WF Id)
GetRule(WF Id)
GetRule(WF Id)
GetRuleStage(WF Id)
GetRuleStage(WF Id)
GetComponents(WF Id)
GetComponents(WF Id)
GetStageTemplates
GetTemplates(Stage)
GetTemplateList(Stage)
GetRuleTemplates
GetTemplates(Rule)
GetTemplateList(Rule)
GetWorkflowsFor Authorization
GetWorkflowForAuthorization (UserId)
GetWorkflows with Status ToAuthorize and UserId <> UserId
AuthorizeWF
AuthorizeWF(WFId, Authorize/ Reject,UserId,Remarks)
AuthorizeWF(WFId, Authorize/ Reject,UserId,Remarks)
GetAllStages
GetWorkflow(WF Id)
GetWorkflow(WF Id)
GetStageAccessForAuthorization
GetStageAccessForAuthorization
GetStageAccessForAuthorization()
SubmitStageAccess
SubmitStageAccess
SubmitStageAccess
AuthorizetStageAccess
AuthorizeStageAccess
AuthorizeStageAccess
GetWFHistory
GetHistory Workflows
GetHistory Workflows along with version numbers
GetRejOrIntWF
GetRejected or IntermediateWF(UserID)
GetRejected or IntermediateWF(User ID)
LogOff
Logoff(UserId)
Logoff(UserId)
DeleteRejOrIntWF
DeleteWorkflow(WFId,UserId)
DeleteWorkflow(WFId,UserId)
IsWorkflowExists
IsWorkflowExists(WFName)
IsWorkflowExists(WFName)
IsWorkflowTemplateExists
IsTemplateExists(Name) Type Workflow
IsTemplateExists(TemplateName) Type Workflow
IsRuleTempExists
IsTemplateExists(Name) Type Rule
IsTemplateExists(TemplateName) Type Rule
IsStageTempExists
IsTemplateExists(Name) Type Stage
IsTemplateExists(TemplateName) Type Stage
IsWorkflowModifiable
IsWorkflowModifiable (WFId)
IsWorkflowModifiable (WFID)
GetRule(Name)
GetRule(Name)
GetRule(Name)

Concepts

XML

The amazing information revolution of the past few years has resulted in tremendous changes in the way we represent, store and exchange data. These changes have brought us to where we are today. However, in the process of trying to deal with the onslaught of information, we have created an overwhelming number of different file formats, standards, and mechanisms for exchanging information. This proliferation of data formats has increasingly complicated our lives as we attempt to interconnect systems that were not originally created to speak to each other. Furthermore, our use of proprietary systems and applications has unpleasantly pigeonholed many of us into trying to shoehorn new uses for old "legacy" systems.

XML hopes to change all this. Stemming from a heritage of structured data formats, starting with SGML and ending most recently with HTML, XML combines the sophistication of structured, metadata-enriched, self-describing data with the ubiquity and simplicity of the Internet. It also presents major advantages over other file format and exchange mechanisms such as EDI, text files, and relational database formats.

Furthermore, the timing is right for XML to take root and grow. The market craves open and nonproprietary systems. The user base of trained developers familiar with HTML, SGML, EDI, and other technologies is growing tremendously. The prevalence and widespread use of the Internet provides a perfect platform for the exchange of information and integration of systems. What 's more, XML has firmly jumped on the positive publicity bandwagon. XML may not have been the first technology of its kind, but it is at the right place at the right time.

What XML is

XML stands for Extensible Markup Language. The basic idea is that with XML, you can encode information in a text document that not only has data in it but also information that describes what the information means and in a structured manner that humans can read.

XML is simply a text document that allows users to store data in a structured manner that also encodes information, or "metadata", as to what that data means.

Asset Oriented Modeling

Modern information systems -- especially the open systems in electronic business and knowledge networks -- require adequate modeling methods. Asset Oriented modeling method is expressive, compact, modular and simple.

The requirements that led to AOM:

- Unified approach to Entities and Relationships.
- Support for higher order relationships.
- Support for complex data structures.
- Namespaces and scopes.

Assets

In traditional conceptual modeling (such as Entity Relationship Diagrams or Object Role Modeling) nouns would end up as entities (or attributes) and verbs would end up as relationships. In AOM both nouns and verbs become Assets.

Assets are the basic building blocks of AOM. Assets can be concrete or abstract. Abstract assets cannot have instances. They can be used for inheritance and type definitions. Concrete assets can have instances. Directed arcs connect assets.

Arcs

An arc connects one asset to another. An arc that points from asset A to asset B indicates that asset B plays a *role* in asset A. Arcs can be clustered.

Cluster

A cluster describes a situation where an arc leads from one asset to a set of alternative assets.

Properties

An asset can have properties. Each property describes a certain aspect of an asset. For example, an asset Person would have properties such as Name, Height, Weight, Birth date, etc. Properties can be atomic or complex.

- Atomic properties can be defined with a simple type expression, or can be defined

- Complex properties are defined via regular expressions (see below), or by specifying a complex type expression.

Constraints

Constraints can be used to define additional restrictions for properties. Constraints can be defined for single properties, across properties, or across assets.

Business Objects & Business Documents

Business Objects are assets or groups of assets that play a role in a business process.

Business Documents are assets or groups of assets that are exchanged as messages between Business Objects within the context of a business process.

Conceptual Modeling

Initially the informal verbal description is put down, and then this description is formalized into a conceptual model. The conceptual model is then transformed into schemas. Identifying the assets and their properties from the informal description arrives at the conceptual model. Anything, which plays a certain role in the context of the business case, is definitely an asset. Arcs then connect the identified assets.

After identifying the assets, properties and arcs, the assets are grouped into business objects and business documents. For each Business object and business document we determine an identifying asset. We then group the remaining assets around these selected assets and form the Meta model.

XPATH

XPath provides a data model and expression syntax for addressing parts of XML documents. X-Query adapts XPath for efficient usage in the database context to fulfill its role as an XML query language. At the same time, it obeys the Tamino design principle of adhering to public standards, as long as there is no official query language recommended by the W3C. (The W3C is currently working on a query language for XML called XQuery.) This XPath-based approach supersedes the previous impiementation of XQL in Tamino 1.x.

XPath is not a query language in itself, rather it is a language for addressing parts of an XML document. Since Tamino generally stores data as XML documents, XPath provides a standard mechanism for addressing any contents of XML objects returned from the database. Beyond the basic expression syntax of XPath, there are other features present in X-Query. With X-Query you can retrieve

- XML objects using query expressions based on the XPath specification, you can retrieve XML objects using the Tamino query expressions that are based on the XPath specification. In order to use these expressions effectively, you will also need to be familiar with the current XPath specification.

- XML objects using Tamino text retrieval and sorting expressions
- In addition to XPath-based query expressions, Tamino also provides text retrieval and sorting expressions which can be used to retrieve and sort XML objects. You can retrieve text by using the special contains operator ~=. You can sort XML objects with the help of the sortby expression. XML objects using Tamino server extensions

You can extend X-Query by using Tamino server extensions.