

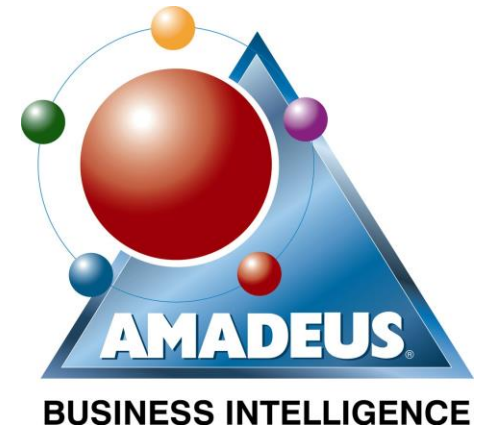


A Successful Implementation of a Complicated Web-based Application Through webAF™ and SAS® Integration Technologies

Clare Nicklin

Amadeus Software Limited

20th May 2003



Introduction

- **Web-based Tax Modelling System**
 - Initially built to run a single tax model
 - Design had to be future proof in order to incorporate and run any number of different models
- **Extendable Design**
 - Final design is metadata driven to enable System Administrators themselves to modify and fine-tune

Application Implementation

- **Software Used and Systems Architecture**
- **Application Characteristics**
- **Development and Technical Difficulties**



Software Used and Systems Architecture (1)

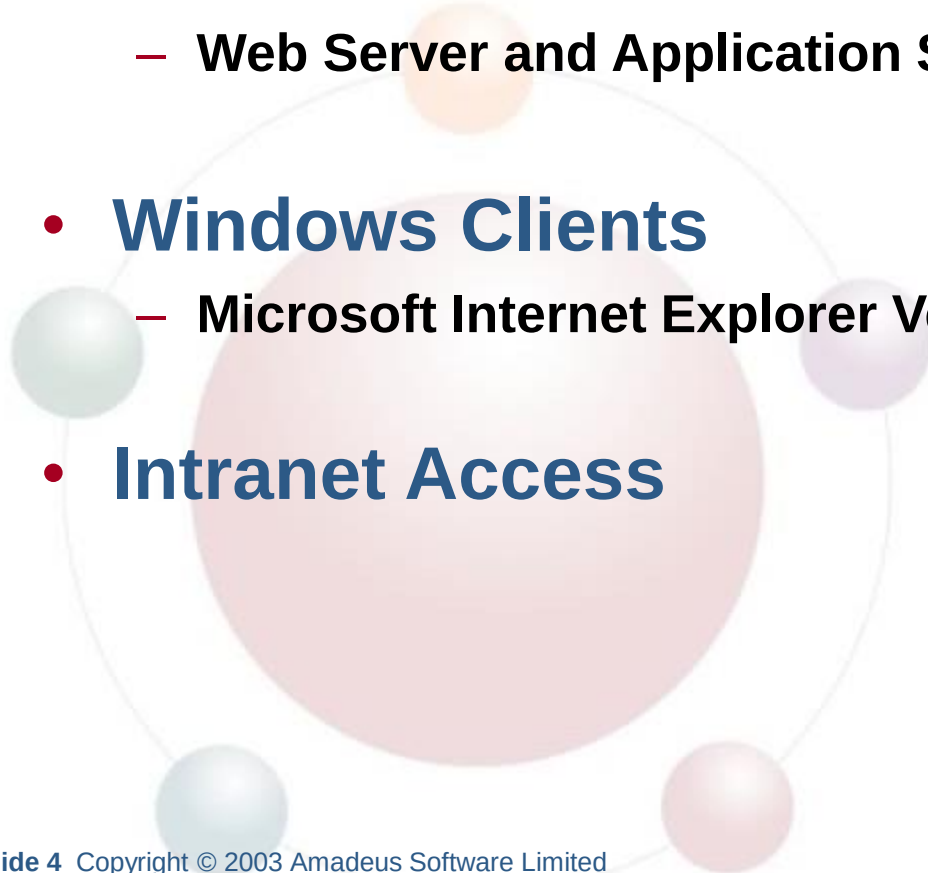
- **Unix Server**

- Web Server and Application Server on same Unix machine

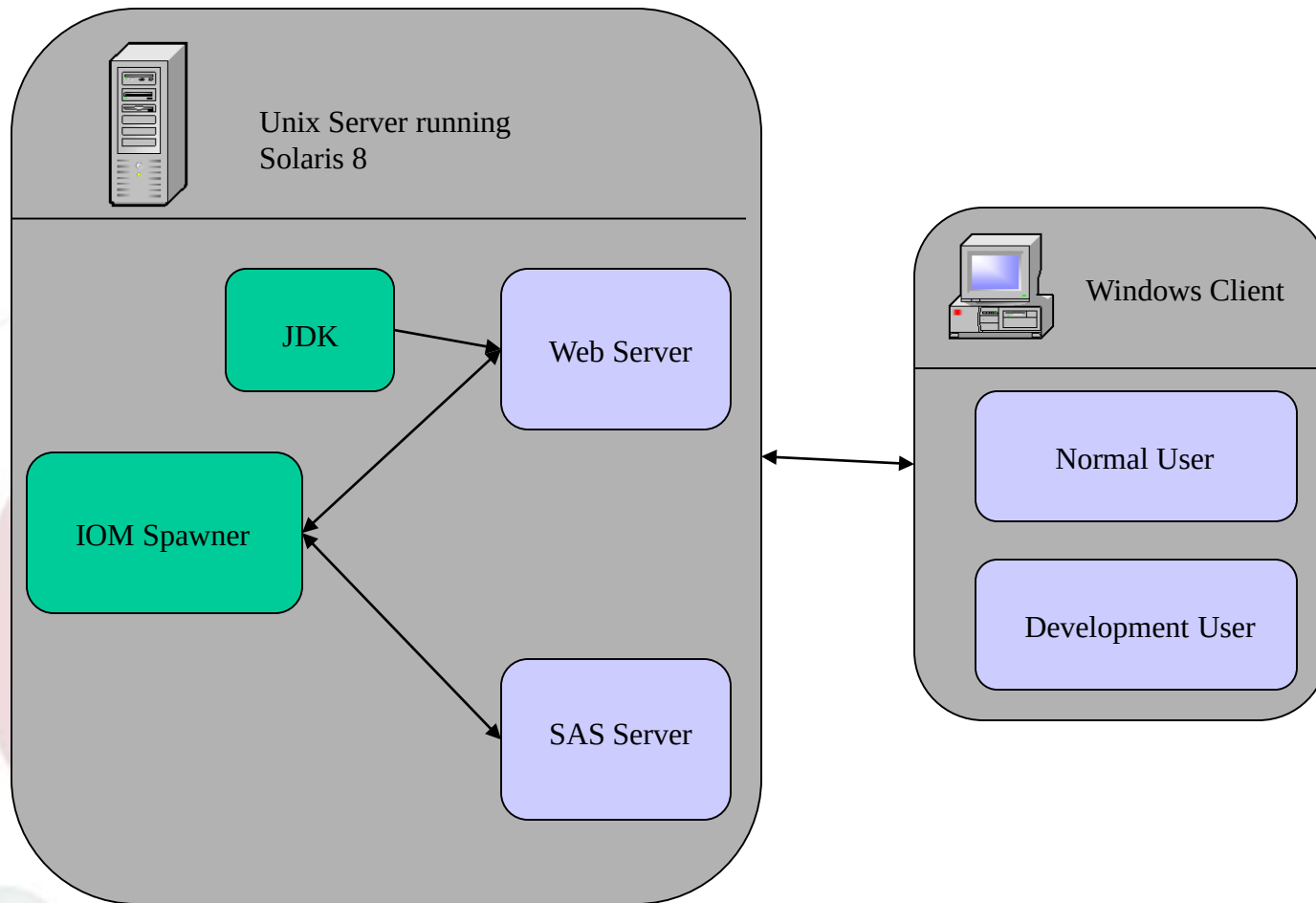
- **Windows Clients**

- Microsoft Internet Explorer Version 5 or above

- **Intranet Access**

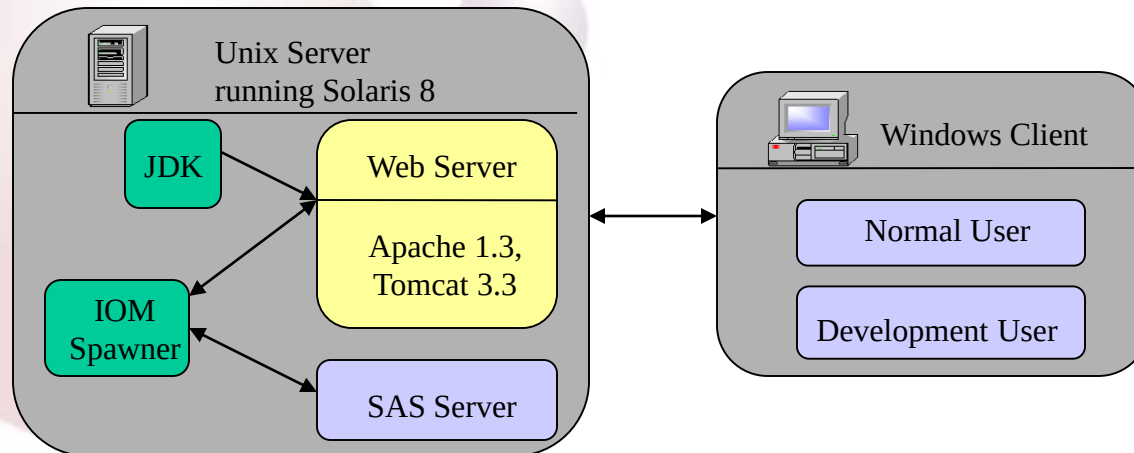


Software Used and Systems Architecture (2)



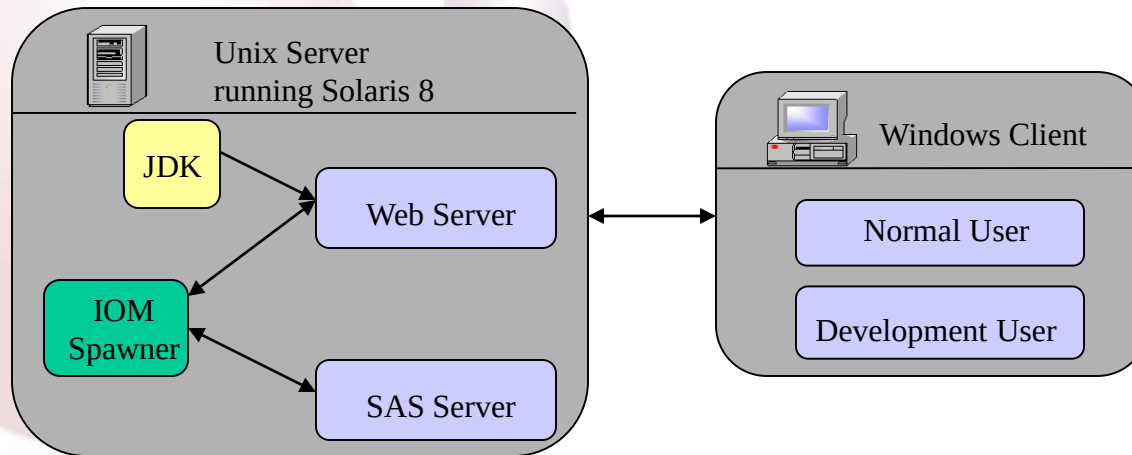
Web Server

- **Hosts web-based files**
 - HTML files, JavaServer Pages (JSPs), other Java classes and JAR files
- **Apache hosts the application**
- **Tomcat (Apache's servlet engine) interprets the JSP's**



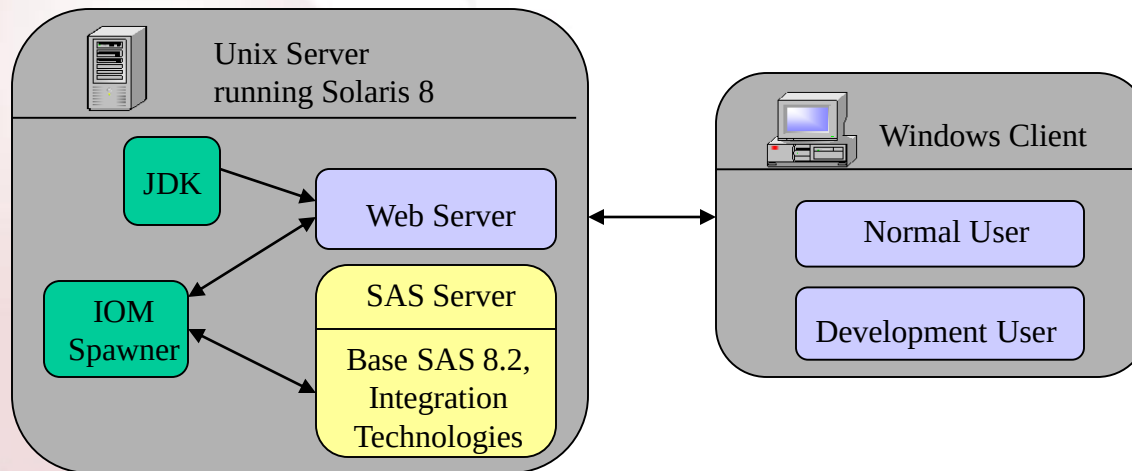
- **Java Development Kit**

- Development tools needed to build and compile Java applications
- Class libraries for execution of programs written in Java
- Additional class libraries in JAR files from AppDev Studio



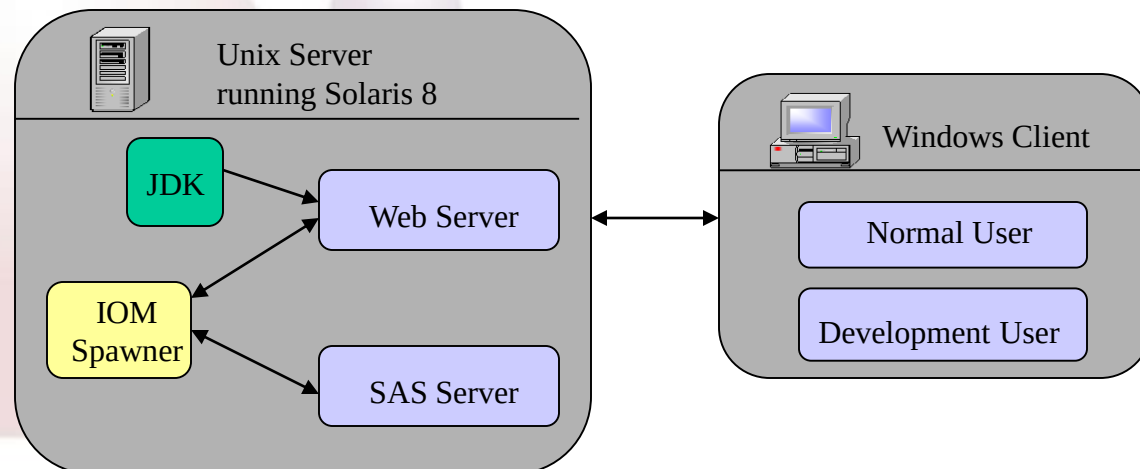
SAS Server

- Contains all data used by system
- All processing carried out in SAS
- Access via Integration Technologies



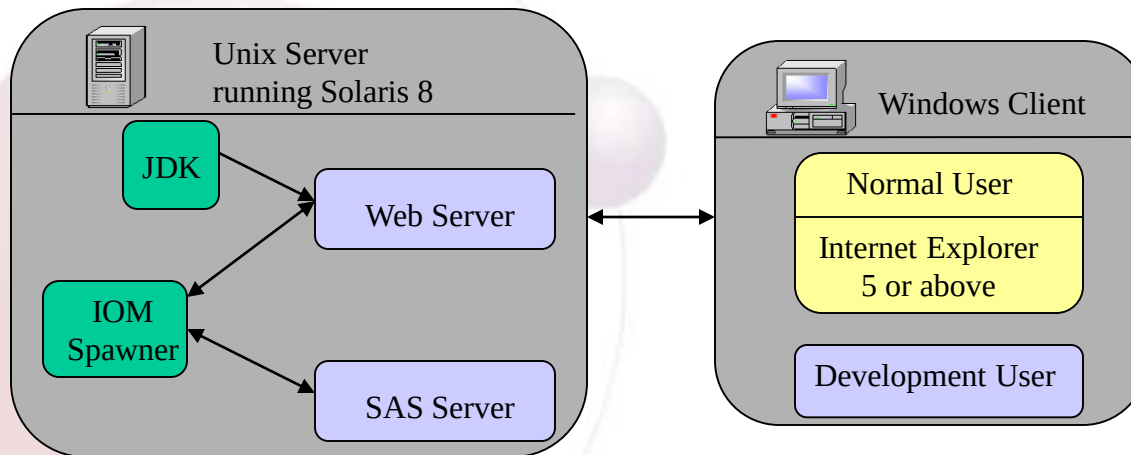
IOM Spawner

- **Integrated Object Model Spawner**
 - Part of SAS Integration Technologies
- **External applications communicate with SAS via the spawner**
 - Listens constantly for calls to SAS on a specific port
- **Handles all communication between Web Server and SAS Server**



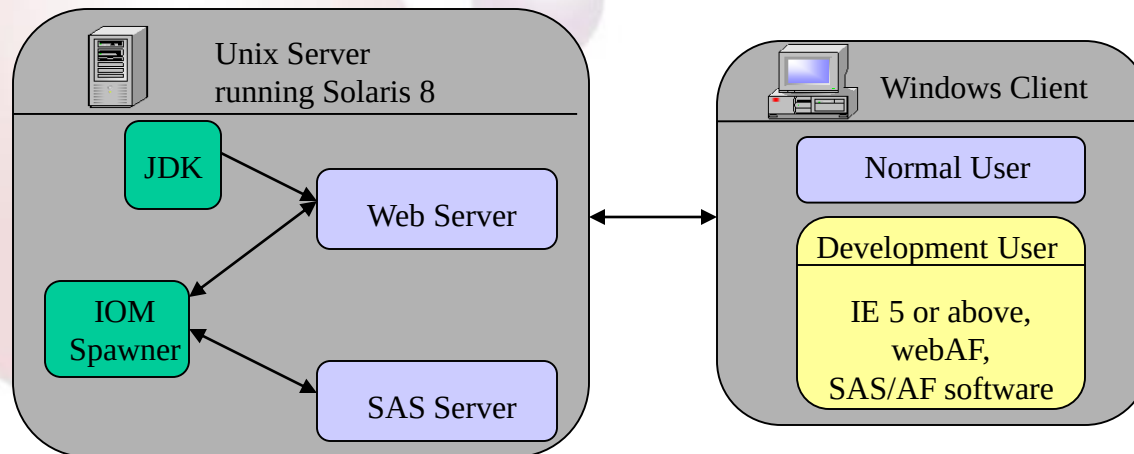
Normal User

- Access to application through browser
- Internet Explorer Version 5 or above



Development User

- webAF used to develop the application
- TransformationBeans used to link to webAF's existing models
- InformationBeans used to access several custom built SCL classes
- SAS/AF used to create those SCL classes



Application Characteristics

- **Metadata Driven**
 - Designed to easily handle future modifications and changes in policy
- **Metadata Managers**
 - Provision for non SAS programmers to edit various sets of metadata
- **Directory Structure Driven**
 - Specific directory structures required to store system data and saved definitions

Metadata Driven

- **Three levels of metadata:**
 - **Environment Level - allows the environment to be changed**
 - i.e. system paths and user logon information
 - **Application Level - administrators can dynamically change the presentation of application screens**
 - Java builds the HTML for the screens content and appearance from the metadata
 - **User Level – stores user related information**
 - i.e. user reporting preferences and user defined formats

Metadata Managers

- **Parameter Metadata Manager**
 - Defines how parameters are displayed on screen. Parameters can be created, edited, moved around etc.
- **Subset Metadata Manager**
 - Defines and registers code and descriptions for subsets of data
- **Stylesheet Manager**
 - Changes the stylesheet used by the system for future proofing design changes
- **Image Manager**
 - Changes the images used in pre-defined locations throughout the system for future proofing
- **User Manager**
 - Users and their privileges can be set up and administered

Directory Structure Driven

- **System Area**
 - Contains code templates, input data, and parameter templates
- **User Area**
 - Contains all user defined jobs and user level metadata
- **Utility Area**
 - Acts as:
 - a recycle bin for deleted jobs
 - a public area for users to share work
 - a store for images used in the Image Manager

Development and Technical Difficulties (1)

- **Browser Timeout**

- Connection timeout errors were received while extensive SAS programs were running
- Due to proxy server timeout halting the connection between the browser and the running program
- Resolved by bypassing proxy servers via the browser settings

- **Rogue SAS Sessions**

- Abnormal termination of the application left SAS processes running on Unix server continuing to take up server resources
- Resolved by tying up the SAS connection with the HTTP session
- HTTP session times out after a preset time period which triggers a stop method on the SAS connection

Development and Technical Difficulties (2)

- **Over-Crowded Screens**

- Many screens need to display large numbers of input fields
- Dynamically building these from metadata whilst keeping them efficient and appealing was challenging
- Resolved by introducing more extensive layout facilities in the Parameter Metadata

- **Speed Issues**

- Information constantly required to be passed between client and server
- Initially tried to use Java rather than SCL methods
- Some Java methods, e.g. retrieving a value from a data set, took up to 4 times longer than the SCL
- Resolved by using more SCL than anticipated

Development and Technical Difficulties (3)

- **Incomplete Installation of Integration Technologies**

- SAS could not find a series of necessary SCL classes
- Resolved by porting the catalogues containing the missing classes from a Windows platform machine

- **AppDev Studio Difficulties**

- Various difficulties encountered generating Java proxies via the InformationBean Wizard
- Memory problems experienced on development machines even though memory exceeded recommended specifications

Conclusion

- **Pushed the limits of Web-based applications using AppDev Studio and Integration Technologies**
- **Metadata driven approach is very demanding in terms of performance and maintenance**
- **However, original business requirements have been met and exceeded**

Questions?

Clare Nicklin

Clare.Nicklin@amadeus.co.uk

Amadeus Software Ltd
www.amadeus.co.uk