Data Presentation, Structures, Formats, and Scalability of Applications - How do They Fit Together?

C.R. Maul

DNRE Tatura, Ferguson Rd, Tatura VIC 3616, Australia (Christian.Maul@nre.vic.gov.au)

Abstract: On the desktop, spreadsheets are a widely used and excellent tool for managing and presenting data. Experience in the use of Excel can be assumed to be common knowledge as it is the predominant spreadsheet program. However, Excel is a purely desktop program and fails to address issues of scalability and distributed computing. This is appropriate for applications with undemanding data requirements. For models with larger data requirements databases are needed and attention must be paid to data structures and data formats. Usually it is not only a technical problem of formats but also an issue of proper data description or metadata. If it is left to the user to produce complex and accurate data in the correct form the application is doomed. Applications start out as simple but grow in complexity and size. Data sets may also grow to a size where databases are required or data generation becomes a specialised task. Formula One (F1) from Actuate is proposed as a solution to all three problems as it is an Excel compatible spreadsheet engine in Java that is capable of both easy data administration and presentation on a small scale, Extensible Markup Language (XML) integration and collaboration with databases for large data sets. F1 allows for both "quick and dirty" implementations and large distributed applications in conjunction with databases while always maintaining user friendliness.

Keywords: Data management; Software engineering; Java; Software components

1. INTRODUCTION

Data formats and scalability usually receive the scantest attention when a new model is designed. Unfortunately this causes problems later on when the model becomes successful is extended. It seems advisable to pay attention not only to the model being developed but also to its possible significance in a larger context. Input and output values and formats play a crucial role from this point of view.

XML seems to be the choice when it comes to data and metadata standards because it is the industry standard and because it is the lowest common denominator between platforms and software systems. However, there are also good reasons not to use it. The effort needed for data production and representation is very high, and components to simplify the job, such as those available from IBM alphaworks (alphaworks.ibm.com), and programming interfaces such as SAX (Simple API for XML) or JAXP (Java API for XML Parsing), are clumsy and still being developed. When using XML life certainly becomes more difficult for the developer as he must provide a means for the graphical representation of the data such as style sheets or a means for data manipulation. Hence, for data description, representation and manipulation purposes spreadsheets seem to be an easy, obvious choice. There are many good table and spreadsheet components available such as JClass Livetable from Sitraka Software (www.sitraka.com), QuickTable from Quicktable (www.quicktable. com), Formula One for Java from Tidestone (www.actuate.com), and VisualSoft JBGrid from VisualSoft (www.visualsoft.com). However, Excel is the predominant spreadsheet program and everyone knows how to use it. Thus, compatibility with Excel in file formats and calculation engine

function becomes an important criterion that only Formula One satisfies.

However, there is also a problem with the distribution of data. Excel is a desktop program. It cannot be used to populate datasets, it does not know which datasets may be available from other users, it cannot administrate different versions of datasets, and it cannot distinguish between redundant and current datasets. To solve these problems we need both a database and a spreadsheet. A database application alone is not an option because we do not want to program a graphical user interface (GUI) for each and every dataset.

The situation is complicated further by the differences of scale in data. Sometimes small tables are needed, sometimes large volumes of data. What can be done to ensure scalability, good graphical representation, easy dissemination and correct data format?

2. METHODS

Four issues need to be addressed:

- visual data representation,
- population of data,
- scalability of applications, and
- metadata.

The solution for our data storage should be distributed, easy to access, have an appealing graphical representation and an obvious display of data structures, and should not require too much programming effort. It seems advisable to use spreadsheets for graphical presentation, but it should be a spreadsheet that can be filled directly from a database or from third party sources. The database should be able to be kept locally or distributed on a server. Very often we do not know the future users of our model, so the ability to scale up the application and connect via the Internet might be important. To minimise the programming effort the spreadsheet should conform to a component standard, have a compact file format and be able to read Excel file formats because this is the predominant spreadsheet program. Metadata that describe data structures are relevant only for a certain size database for which the number of attributes has become confusing and when the database structures must be communicated.

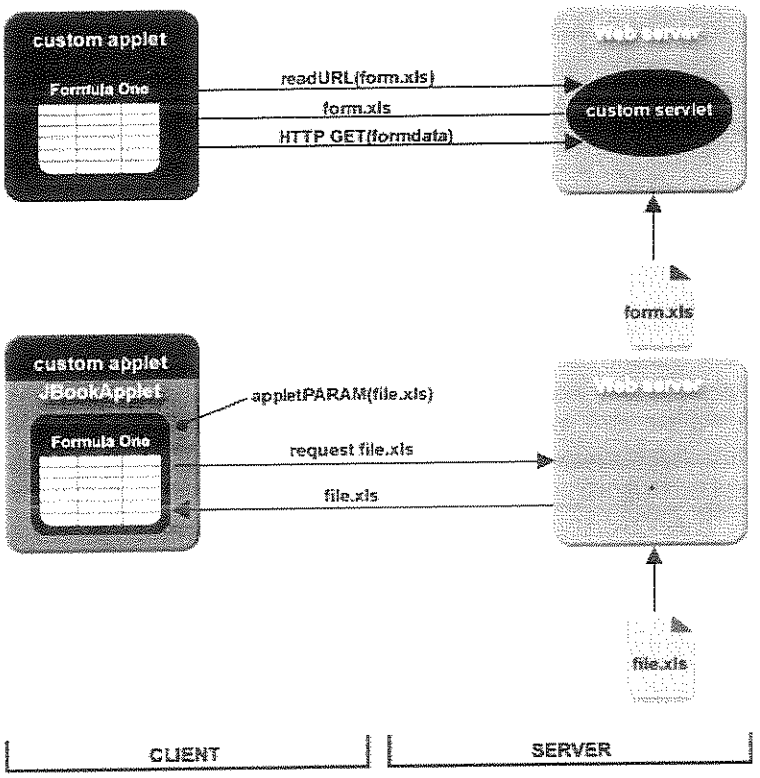

Figure 1. Using F1 with a central data repository (modified graphic from Actuate, http://www.tidestone.com/demos)

3. RESULTS

The solution to these requirements could be a component such as Formula One from Actuate. Actuate calls it an 'API-driven spreadsheet engine'. It can read and write Excel formats, generate XML or HTML (Hypertext Markup Language), can connect to JDBC databases (usually translated as Java Database Connectivity, although it is not an acronym according to Sun), [Flanaghan et al. 1999] and has graphics capabilities. It provides a GUI for data display and Excel compatible formulae (without the well-known errors in Excel's statistical formulae) if the data need to be manipulated.

Because it is a pure Java component it can be used in standalone programs or in applets. Apart from the reporting functionality F1 enables integration into networks services through either the Java Server Pages (JSP) API, servlets which are programs that reside and execute on a server, or the Java 2 Enterprise Edition programming interfaces (J2EE). F1 provides a GUI in the simplest case which is a standalone application and data in the form of Excel files.

However, if this does not suffice and data need to be kept centrally, it is possible to go to the next step, which is a client-server structure. Data from a web server are loaded from a file or Excel-form into an applet as shown in Figure 1. Likewise a Java application could access a file via a servlet or directly from a web server.

The F1 component itself resides in the client application in all scenarios so far.

If the quantity of data grows different scenarios are possible as shown in Figure 2. The data can be obtained from a database and transported to the client in one of four formats: XML, HTML, F1 or Excel spreadsheet data. The XSLT (Extensible Stylesheet Language Transformations) processor uses an associated XSL document to filter and format an XML file into an F1 spreadsheet on the server. The populated template is then calculated, processed and passed on to the browser via the servlet's output stream.

Using a spreadsheet database as input requires no metadata because this is an appropriate solution only for smaller quantities of data. Using XML there is no metadata issue because XML comes with its own metadata standard which is a description of the data and their structure in the file header.

The model would usually be part of the servlet. In the case of XML and HTML data there is no way to run any model locally as it must be part of the servlet that sends the results to the client.

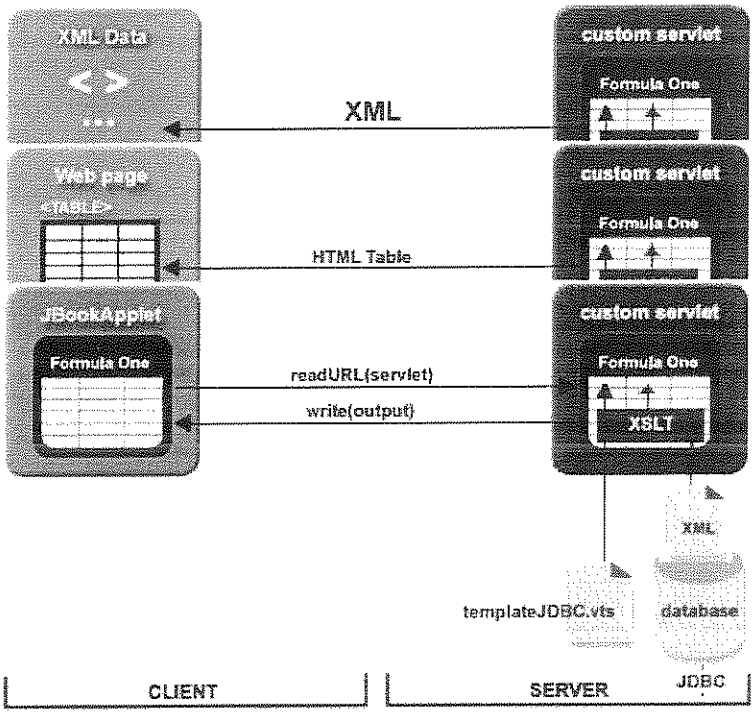

Figure 2. Using F1 with a central database (modified graphic from Actuate, http://www.tidestone.com./demos)

The model could just as easily sit on the desktop as a stand alone application that works on data tables or records received via JDBC. The java.sql.Connection object does not distinguish between local and remote databases. It expects a location to be defined as a URL. However, it is necessary to transform the data received into something F1 can read. This might be XML with an XSL stylesheet, Excel or F1 files, SQL data which are mapped to objects or variables or, in the worst case scenario, tab or comma delimited flat file data which are error prone.

If serious resources and multiple simultaneous access are required even Java 2 Enterprise Edition (J2EE) solutions are possible with middleware and application server. The scarce resource when using this solution is more likely to be knowledge rather than money because there are open source application servers available such as JBOSS (www.jboss.org) or JOnAS (www.evidian.com).

The client side is as flexible as the server side: data could be delivered as XML or HTML to the client, or as an applet with F1 doing the display, or the servlet could use Excel to display by setting its output stream with a Multipurpose Internet Mail Extension (MIME) type to Excel. This forces the browser to load the Excel plug-in and display the spreadsheet in Excel on the client's machine.

Another alternative, displaying the data with an applet, enables the user to run the model both either locally, as part of the applet, or remotely, where it is embedded in the servlet. The model can run entirely or partly in the applet because the entire range of the Java programming language is available on both sides.

In all other cases the model must run in the servlet. F1 resides only on the server side unless it is used for display purposes on the client side.

Obviously, the design of the application on the client side determines the server side. In other words any processing, that is running of a model, that cannot or does not happen on the client side must happen on the server side and vice versa.

4. DISCUSSION

Using a component like F1 helps to quickly develop a good looking GUI and to get data into any application easily. It achieves a good deal of our target of making life as easy for the developer as it is or should be for the end-user. The presentation of data as spreadsheet data enables comfortable data manipulation and builds on the existing knowledge of model users.

What is more, as data sets grow they can be kept centrally in a database. The design is scalable to any size to which our model might grow in the future. It can consist of spreadsheet files on a desktop and it can adapt to large, diverse structures such as webserver with JSP with or without servlets and database or J2EE applications with middleware and application server. However, it enables the developer to stay on any step of the ladder if it is sensible and comfortable and to move on to the next more complicated step should it be required. At any level it looks the same for the end user regardless of whether his objects are local or remote ones. The beauty for the programmer is that everything done so far can be reused while adding the complexity needed to run distributed applications.

If the system is located on a server no client-side software or client data need to be updated.

Using components such as these does come at a price. Developing programs this way can be quick but it is certainly expensive. However, because all future development opportunities are kept open it is a worthwhile expense for projects with a long term perspective.

5. REFERENCES

Flanaghan, D., Farley, J., Crawford, W., Magnusson, K., Java Enterprise in a Nutshell. O'Reilly, Sebastopol, CA, US, 1999.

http://www.tidestone.com/demos_f1j_9.0/about.js F1 for Java 9.0 - Java API Demos1999

For each of these options, the cane supply optimisation (CSO) model optimised decisions relating to the amount (tonnage) of sugar cane to be harvested from farm paddocks by crop class and variety, during fixed time intervals (fortnights) throughout the harvest season. These decisions were generated according to the criterion of maximising net profit to the sugar industry, within a sugar mill region, while meeting the operational constraints associated with harvesting, transport and mill crushing capacity [Higgins, 1999]. As net profit is linked to a relative payment scheme where a grower is paid according to daily calculations of farm CCS units relative to the mill CCS for cane supplied on the day, the CSO model also requires input data which describes the trends in farm CCS relative to the mill across the harvest season. Using historical block productivity data [Chardon and Smith, 1993], the estimation of farm and paddock relative CCS was achieved through the development of a statistical model which captures some of the variation associated with farm (confounding the effects of farm management and physical location), crop class and variety, with harvest date [Higgins and Haynes, 2001]. A crop class refers to whether the cane on a farm paddock is plant crop (around 1 year after planting), first ratoon (around 2 years after planting), second ratoon and so on. A ratoon is the re-growth of the plant from the established plant base (stool) after the top is removed at harvest.

The application of these options analysis techniques have shown considerable potential gains to the industry for case studies in the Mossman, Mackay and Maryborough sugar regions [Higgins, 1999]. For the potential gains to be wholly or even partially realised in practice, the optimal harvest date decisions generated by the CSO model for the chosen harvest scheduling option had to be presented in an uncomplicated form that can be successfully implemented in the field. Therefore, the implementation phase of the project required the formulation of a framework for generating guidelines which provided a suitable harvest schedule for the clients (growers, harvester operators and millers) to implement in the field while reflecting the optimal harvest date decisions required to achieve the gains estimated by the CSO model.

The large numbers of growers interested in participating in the pilot studies for the 2000 harvest season, highlighted the need for the fast and efficient generation of harvest schedules for selected farms. This was achieved through the development of a framework which incorporated:

- the CSO model,
- a database system which combines model results with paddock crop information and stores historical productivity data,
- decision support tools, and
- expert intervention by the grower.

The whole process involved in establishing the pilot studies, including research into appropriate model development, extensive consultations with growers and the development of a framework for generating harvest schedule guidelines, has provided new tools and a better understanding of the requirements for the successful implementation of alternative cane supply options.

2. PILOT STUDIES OF THE CANE SUPPLY OPTIONS

During the 2000 and 2001 harvest seasons, pilot studies were conducted in the case study regions of Mossman and Mackay, and in Maryborough during 2001, to evaluate the success of the framework in producing harvest schedules which are easy to implement in practice, and also to assess the validity of the harvest decisions generated by the CSO model. The evaluation of these pilot studies for the 2000 harvest season can be found in [Higgins et al., 2000].

The industry agreed to pilot Options 2 and 3 in Mossman, Mackay and Maryborough. In addition, Option 1 was piloted in Mackay through the Harvesting Equity Exchange Scheme (HEES) [Muchow et al., 2000]. Option 2 presents an alternative cane supply arrangement for harvesting groups that prefer not to participate in full geographical harvesting or are not suited to Option 1. Participation in Option 2 involves the relaxation of farm equity within a group and may lead to a potential gain in farm profitability without sacrificing harvesting group equity.

Option 3 provides growers with guidelines to make informed decisions about best harvest dates of farm blocks without any change to the existing harvesting and farm equity structures. The greatest benefits are most likely to be realised for farms which have geographically dispersed blocks with different crop class characteristics. All growers not participating in Option 1 were invited to participate in the pilot study for Option 3.