A User-Driven Environment for Financial Market Data Analysis

Fethi A. Rabhi[1], Omer F. Rana[2], Adnene Guabtni[1], and Boualem Benatallah[3]

[1] School of Information Systems, Technology and Management, UNSW, Sydney 2052, Australia
f.rabhi@unsw.edu.au
[2] School of Computer Science, Cardiff University, Cardiff CF24 3AA, UK
o.f.rana@cs.cardiff.ac.uk
[3] School of Computer Science and Engineering, UNSW, Sydney 2052, Australia
boualem@cse.unsw.edu.au

Abstract. This paper proposes a software development environment which facilitates the analysis of large financial datasets by end-users. This environment is based on an event-based data model that gives a coherent representation of market activities, particularly high-fequency market and news data. The model makes it possible to define software components and Web services to manipulate entities in the model. The paper also describes a prototype implementation which allows domain experts to compose components and services to build an application. This prototype uses the Triana scientific workflow system to define workflows of existing software components and Web Services. This approach is demonstrated on a realistic case study related to processing both news and financial market data.

Keywords: financial data, Triana, scientific workflow, timeseries analysis, news analysis, event model, composition framework, Web Services.

1 Motivation and Introduction

With the dramatic increase in the speed and availability of computer networks, a significant proportion of all economic activities are now conducted electronically. In particular, the field of financial trading has seen an unprecedented increase in the number of participants and the volumes of trades conducted via electronic markets [27]. As a result, high frequency data has become increasingly available for historical analysis by researchers in fields as diverse as econometrics, finance and accounting. Datasets are often stored in a format suitable for viewing as a spreadsheet. A row usually corresponds to a timestamped piece of information such as the occurrence of a trade, a variation in an instrument's price or an index, the publication of a news story or a market announcement etc. Our research focuses on datasets originating from financial exchanges made available by third party information providers to the research community. For example, SIRCA's Taqtic system [17] provides users with a Web interface for downloading market

D. Kundisch et al. (Eds.): FinanceCom 2008, LNBIP 23, pp. 64–77, 2009.

data according to search criteria such as type of instrument, exchange, time period or frequency (e.g. intraday or interday data). There are several similar portals offering users the possibility to download financial market data (e.g. WRDS [25]).

Analysing such datasets requires expert domain knowledge (e.g. in finance and microeconomics), experience and IT skills [26]. Besides being able to identify suitable data sources and specify the right search criteria, users must be able to perform a wide range of analysis functions (statistical, data mining, language processing) and present results in a suitable form (e.g. through visualisation or report creation). Analysis processes cannot be determined in advance as users tend to perform tasks in a piecemeal fashion: they use a dataset to generate some results, they combine results to build new datsets and the process may be repeated iteratively with datasets obtained with different search criteria (e.g. a different time period). Users also tend to use a variety of tools such as Excel and Matlab to store results and perform routine calculations. When the type of analysis is complex, users spend a lot of effort cleaning, reading and interpreting the data, converting datasets from one format to another, copying some results from one file to another, merging datasets with different semantics, which increases analysis time and the risk of errors.

Building software tools to support such analysis processes is a very challenging task for a number of reasons. Firstly, it requires a deep understanding of financial market operations which skilled software engineers do not possess. Secondly, as data formats vary considerably and little standardisation has taken place in this area, an automatic tool that guarantees information consistency between the different stages of the analysis process would be difficult to develop. Thirdly, it is always preferable to let domain expert users be in control of the analysis process. According to David Shirref, "humans can understand intuitively a great deal about markets. They can scan the complex factors governing a market far more efficiently than any computer"[16].

This paper describes an approach which facilitates the definition and execution of analysis processes by proposing a conceptual model which unifies different views of the datasets, then offering high-level services for manipulating datasets according to these concepts, and finally giving the user the possibility to compose such services in a way that addresses their requirements. It is organised as follows. In section 2 we describe an event-based data model to give a coherent representation of market activities. In particular, we have developed a model that adequately represents high-fequency market and news data. The model makes it possible to define software components and Web services to manipulate entities in the model. Some services will be responsible for querying market and news data from existing repositories (thus acting as event sources), some will implement event processing functions (e.g. filtering, aggregation), some will determine relationships between events (i.e. event patterns) etc. Section 3 contains a case study in which complex analysis business processes can be decomposed into a hierarchy of services. In section 4 we describe a prototype implementation consisting of a user-driven composition framework that allows domain experts to

combine components and services to build an application. The prototype allows for incremental development, thereby giving users the possibility to refine their application in an iterative fashion.

2 A Conceptual Model for Analysing Financial Market Data

In our conceptual model[9], electronic markets (e-markets) can be thought of as distributed event systems, consisting of market events – which capture data attributes such as bid/offer, types of products being traded, volume/number of products, etc; and news events - which capture data related to particular news stories – as published by news organizations such as Reuters. An e-market system may operate in different phases – pre/post-trade and trading. The pre-trade phase involves the submission of buy/sell offers to the market, with the market generating trades during the trading phase. The post-trade phase involves analysing the trades that have taken place, and understanding whether market rules have been followed, and signalling any illegal behaviour. Each of these phases generates different types of market events, and produce different event patterns. In this paper, we are only considering events originating from exchanges, as they represent the type of datasets that are widely available for researchers.

2.1 Basic Event Concepts

In our conceptual model[1], an event of type E which occurs at time t is denoted $e^{(t)} : E$. Each event has a collection of attributes associated with it, hence $e_j = \{a_1, ..., a_m\}$, where each a_i is an attribute with one or more values. An event may be uniquely characterized by its attributes and the time at which it has been recorded. We also consider that the occurrence of such an event is non-deterministic, with no pre-assumed distribution associated with when an event is generated. Our event detection/recording mechanism starts at some time t_0, and then a sequence of events (of type $< E >$) at a given time point $s^{(t)}$ may be specified as:

$$s^{(t)} = e_1 \triangleleft e_2 ... \triangleleft e_n :< E >$$

where e_1 represents the first event that was recorded when the event detection process began, similarly e_n is the last event that was recorded before the current time point t_n. The relation $\triangleleft$ represents an ordering on the recorded events. We make no assumption about when the event was generated, and ordering between events is based on the detection/recording of these events by our system – i.e. when the event was detected. In this context, two events e_i and e_j may have the same attributes and values, but will be treated as different events if they are recorded at different times. Any time stamp on the event itself (as part of its

[1] $x : T$ denotes that an object x is of type T.

attributes) is not used to order events – as time mechanisms (clocks) on external systems that generated the events cannot be guaranteed to be synchronized. In the simple case, we may assume that all events are recorded by the same system, and therefore the time stamp on these events may be used to order them. However, when events may come from different systems, it may not be possible to assume any pre-existing event ordering. Given this representation, we define a grouping of events as:

$$\{e_1 \triangleleft ... \triangleleft e_k\} : \{< E >\}, k \leq n$$

where each group is represented by a sequence of events that has a particular semantics. For instance, each sequence in a group may represent those events within the original sequence that correspond to a "price jump". As an another example, one may consider a group to represent the changes that take place in a particular news story – where all events in such a sequence have some common attributes defined through some ontology. More information about the types of events and their attributes in a financial market context can be found in [9].

2.2 Event-Based Services

Assuming these basic concepts, any type of financial data processing capability can be thought of as a service that operates on events. Frequently used services fall under the following categories[2]:

- **Event Provision Services**: given a query object (of type Q), and the location of an event source (of type S), these services extract a sequence of events that match the constraints identified in the query:

$$EventQueryST \ = \ L \times Q \ \rightarrow \ < Event >$$

- **Event Processing Services**: given a sequence of events, process them to generate another sequence of events. Examples of processing are adding/ removing events, enriching existing events, aggregating events, etc. Some possible services in this category are:
  - *Event enrichment services*: given a sequence of events, enriches the attributes of these events with new information (either supplied from an external source or derived from other attributes)

$$EventEnrichST \ = \ < Event > \ \rightarrow \ < Event >$$

for each event e_j (for all j in the input event sequence) that has attributes $\{a_1, ..., a_m\}$, such a service extends this to $\{a_1, ..., a_m\} \bigoplus \{a_1^{'}, ..., a_m^{'}\}$. The $\bigoplus$ operator involves extending the values associated with one or more existing attributes, and does not involve the addition of new attributes to the existing set $\{a_1, ..., a_m\}$.

[2] $T_1 \times \ldots \times T_n \ \rightarrow \ R$ denotes the type of a function with n arguments of type $T_1 \ldots T_n$ and whose result type is R.

- *Event pattern detection services*: given a sequence of events, identify groups of events that match some user-defined characteristics.

$$EventPatternDetectionST \; = \; < Event > \; \rightarrow \; \{< Event >\}$$

  each group produced as a result is an event pattern instance i.e. an event sequence that matches the pattern being detected. Also, if e is an input event sequence, and p is a pattern instance, we have the property $length(p) < length(e)$.
- *Event classification services*: such a service groups a sequence of events in several classes according to some classification criteria.

$$EventClassificationST \; = \; < Event > \; \rightarrow \; \{< Event >\}$$

  each group produced as a result contains events that belong to the same class according to the classification method selected.

– **Reporting services**: refer to services used to convert event streams or event patterns into a format suitable for text display, download or visualisation.

$$PresentationST = \{< Event >\} \; \rightarrow \; (String \mid HTML \mid Graph)$$

A service implementation can be supported by a local component (i.e. the executable code and data are held on the same machine), or can be remotely invoked – using a Web Service interface. Remote execution would involve transferring object instances from our model to a remote machine. In most cases, existing software modules/packages or Web Services can be adapted through the use of service wrappers.

3 Case Study

This case study demonstrates how researchers have the possibility to define their own analysis processes from a community of services that have been designed according to the conceptual model described in the previous section. At the highest level, researchers are interested in understanding how market events may be influenced by news events, although the exact relationship between them may not be known *apriori*. Market events may be limited in their type, whereas news events may be categorized based on the particular domain of interest to a user.

In this case study, we assume that the researcher is interested in one particular asset over a fixed period and that all market and news data related to this period is available. The analysis process is conducted in two distinct stages respectively called *price jump detection* and *news summarization*, each stage containing several variations. In the rest of this section, we describe each analysis stage (and its variations) as an application of different types of services.

3.1 Price Jump Detection

In the first stage, asset prices are analysed over the period of interest, looking at *exceptional* fluctuations in prices or *price jumps*. Being able to distinguish between jumps and continuous sample path price movements is important as it has implications for risk management and asset allocation [3]. Determining when a price jump has occurred is influenced by the volatility of the market, and the type of asset being considered. It is possible to utilize statistical approaches, based on a time window of recorded asset prices (using the quadratic variation process [1]), to calculate jumps in asset prices. Much of this work relies on the assumption of a continuous price process – whereas in our work we are primarily considering a sampling of this process through market events. Our approach therefore approximates this continuous market process through a discrete time process – generally through ordered events.

Price jumps can be measured in a number of possible ways including measuring sums of squared returns over some time interval [1], use of central limit theorem for realized variances in prices, use of parametric models, use of non-parametric, Markovian analysis and Monte Carlo based finite sampling. Researchers select an appropriate technique depending on the state of the market at a particular time and their particular expertise. In some cases, researchers need to experiment with different techniques before finding a suitable one. Our model supports "plug-and-play" experimentation as each technique is supported by a component of type $EventPatternDetectionST$ (thereby allowing a user to analyse the same data using different algorithms concurrently):

$$priceJumpDetectionService : EventPatternDetectionST$$

Each sequence of events returned by the service corresponds to a price jump determined according to the technique implemented. Each technique may have additional parameters specified by the user.

3.2 News Summarization

Price jumps are, in the first instance, only used to identify particular time periods of *interest* that require further analysis. In the second stage of the analysis process, the researcher tries to identify which categories of news have had an impact on prices. There are many techniques that can be used to analyse and group news stories. In this case study, we assume researchers want to use summarisation techniques [23]. Given news events that are tagged with specific keywords, event summaries are created by grouping together events with similar tags according to a user-supplied ontology. In our model, a summarization technique is supported by a component of type $EventClassificationST$:

$$summarizationService : EventClassificationST$$

Each sequence of events returned by the service represents news events with "similar" tags. The summarization service relies on an ontology – therefore the choice of an ontology impacts the obtained results.

To be effective, the summarization service assumes that appropriate tags and ontologies have been defined and applied to the news events sequence supplied as input. One key theme in the current project is also to investigate how ontologies can be used to provide news summaries at different levels of granularity. Significant work already exists about tagging news events, such as NewsML [11], Semantic Web related efforts such as "Calais" (part of the Sekt project) [15] and work by Sanchez-Fernandez et al. [14]. There are also existing efforts that try to identify connectivity between news stories, and to visualize the outcome [12]. Our model captures these variations in the form of event enrichment services. A technique to add tags to news events is supported by a component of type $EventEnrichST$:

$$computeNewsTagsService : EventEnrichST$$

Therefore, there are many variations in the way news summaries are determined. Based on the output, researchers have the choice between using different ontologies, tag generation schemes and summarization techniques. In the implementation described next, we illustrate one possible analysis scenario.

4 Implementation

In this section we describe a prototype implementation using the Triana [20] scientific workflow system. To address the requirements of the case study, a number of software components have been developed and assembled in a way that expresses various analysis processes through Triana. We first provide a general introduction to Triana followed by a description of each supported process.

4.1 Using Triana as a Service Composition Tool

Triana was initially developed by scientists in GEO 600 [7] to help in the *flexible* analysis of data sets generated from scientific instruments, and therefore contains many of the core tools needed for one-dimensional data analysis, along with many other toolboxes that contain units for areas such as signal processing (e.g. FFT, Spectral analysis) and text processing (e.g. string comparison). All in all, there are around 500 units within Triana covering a broad range of applications. Further, Triana is able to choreograph distributed resources, such as Web Services, to extend its range of functionality. The Web Service-based algorithms have also been added to Triana to support data mining [18]. Triana may be used by applications and end-users alike in a number of different ways [19]. For example, it can be used as a: graphical workflow composition system; a data analysis environment for image, signal or text processing; as an application designer tool, creating stand-alone applications from a composition of components/units; and through the use of its pluggable workflow representation architecture, allowing third party tool and workflow representation such as WSDL and BPEL4WS. A workflow graph in Triana is encoded in XML, and can be mapped to a BPEL representation, for instance.

The Triana user interface consists of a collection of toolboxes containing the current set of Triana components and a work surface where users graphically choreograph the required behaviour. The modules are later bound to the services that they represent to create a highly dynamic programming environment. Triana has many of the key programming constructs such as looping (do, while, repeat until etc.) and logic (if, then etc.) units that can be used to graphically control the dataflow, just as a programmer would control the flow within a conventional program using specific instructions. Programming units (i.e. tools) include information about which data-type objects they can receive and which ones they output, and Triana uses this information to perform design-time type checking on requested connections to ensure data compatibility between components; this serves the same purpose as the compilation of a program for compatibility of function calls.

Triana has a modularized architecture that consists of a cooperating collection of interacting components. Briefly, the thin-client Triana GUI connects to a Triana engine (Triana Controlling Service, TCS) either locally or via the network. Under a typical usage scenario, clients may log into a TCS, remotely compose and run a Triana application and then visualize the result locally – even though the visualization unit itself is run remotely. Alternatively, clients may log off during an application run and periodically log back on to check the status of the application. In this mode, the Triana TCS and GUI act as a portal for running an application, whether distributed or in single execution mode.

4.2 Specifying a Price Jump Detection Process

The first process developed consists of detecting price jumps. To extract market data, we developed a Web Service called `MarketQueryService` that returns market data (e.g. trades, quote and market depth) according to a number of criteria specified in a (`MarketQuery` : Q) object:

$$\texttt{MarketQueryService} : \mathit{EventQueryST}$$

Generating a query object through the graphical user interface is achieved through the `MarketQueryGen` component. (`DataSetLocation` : L) is a component that supplies the source of market data but this is restricted to SIRCA's Taqtic system in this prototype. The market query service returns a stream of events from the specified source that correspond to the query search criteria.

We also developed a `PriceJumpDetection` module in Java which implements a simple time series analysis based on conditional forecasting. The module makes use of the times series properties of the price series based on the estimation of an autoregressive model. For an autoregressive polynomial of order 1, the process reduces to a simple first order autoregression in the form,

$$y_t = c + \phi y_{t-1} + \epsilon_t$$

ϵ_t is the part of the dynamics not captured by the information available to the user, i.e., all past values of y and other variables. In this model ϕ, c are

unknown to the researcher. We estimate these parameters using either ordinary least squares or maximum likelihood estimation[8]. Given these estimates, we define a *Jump* as any price value above a number of standard deviations from the process long run estimated mean $\widehat{\mu}$. This is similar to constructing confidence intervals around the mean and characterizing the prices outside this confidence region as potential jumps. The number of standard deviations is determined by the probability tails of a *standard normal* distribution. For example, a 95% confidence interval corresponds to 1.96 standard deviations. We used both 90% and 95% levels as they are standard values among researchers. However, the user can vary this parameter and the module can be adapted to internally calculate the length of the confidence interval. The algorithm based on the unconditional moments is as follows, for $t = 1, .., n$

$$if\ y_t < \widehat{\mu} - z_{\alpha/2}\widehat{\sigma}_y\ or\ price_t > \widehat{\mu} + z_{\alpha/2}\widehat{\sigma}_y\ then\ Jump$$

where $z_{\alpha/2}$ is the number of standard deviations associated with $(1-\alpha)100\%$ confidence level. We performed simulation where we generated a series with an autoregressive structure of known order. We then injected into the series a number of jumps in the price. The module did detect all the jumps that are outside the confidence intervals and outside the forecast intervals.

$$\texttt{PriceJumpDetection} : EventPatternDetectionST$$

Different algorithms can be built to account for different time series dynamics. Models that account for unobservable price changes using *ordered Probit model* specifically cater for high frequency data. Duration models are more concerned with the time interval between trades to account for intraday activity that is missed in the standard low frequency time series modeling (see Tsay [24] for a thorough exposition).

Finally, the components `PJString` and $\texttt{PJVisualiser} - \texttt{HTML}$ allow the results to be visualised in comma separated values (CSV) or HTML format respectively. Figure 1 illustrates the final price jump detection process expressed in Triana.

4.3 Specifying a News Analysis Process

Before news can be analysed, they have to be extracted from the archive. We developed `NewsQueryService` as a Web service that extracts news according to a number of criteria specified in a $(\texttt{NewsQuery} : Q)$ object.

$$\texttt{NewsQueryService} : EventQueryST$$

As previously mentioned, this service takes a $(\texttt{DataSetLocation} : L)$ as an argument but this is fixed in our prototype (SIRCA's news archive only). The news search query is constructed from the results of the previous workflow using the `PJ2NewsQuery` software component. The user is offered different options in filtering the price jumps that are of interest through the `PJFilteringOptions` component.

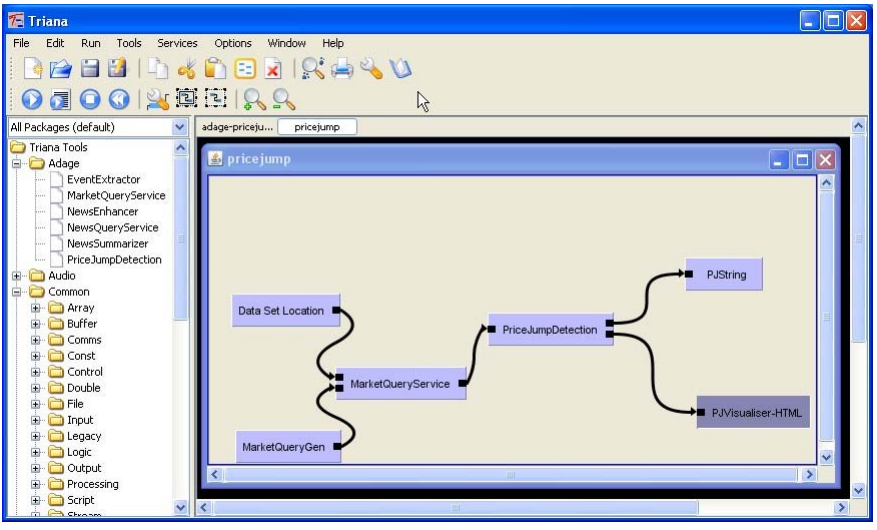

Fig. 1. Price Jump Detection Workflow

The news query service returns news events with very limited tags (Reuters topic codes only). For this reason, we developed a `NewsEnrichment` module which extends the tags of each news event by new tags extracted from the news story. This is based on a simple text analysis technique. In the future, we plan to integrate external Web services such as Thomson-Reuters' Calais [21] to provide a richer (and more appropriate) set of tags.

$$\texttt{NewsEnrichment} : \mathit{EventEnrichST}$$

The `NewsSummarizer` module builds news summaries according a technique proposed in [13]. The ontology is specified in an `OntologyLocation` object which can be modified by the user through an external tool (Protégé [22] in this prototype).

$$\texttt{NewsSummarizer} : \mathit{EventClassificationST}$$

Finally we developed some event presentation modules to visualise streams of events or event patterns:

$$\begin{aligned}
&\texttt{PJVisualiser} - \texttt{HTML} && : \mathit{PresentationST}\\
&\texttt{PJString} && : \mathit{PresentationST}\\
&\texttt{NewsSummariesCSV} && : \mathit{PresentationST}\\
&\texttt{NewsSummariesViewer} - \texttt{HTML} && : \mathit{PresentationST}
\end{aligned}$$

Figure 2 show the previous process with the news summarization process added to it.

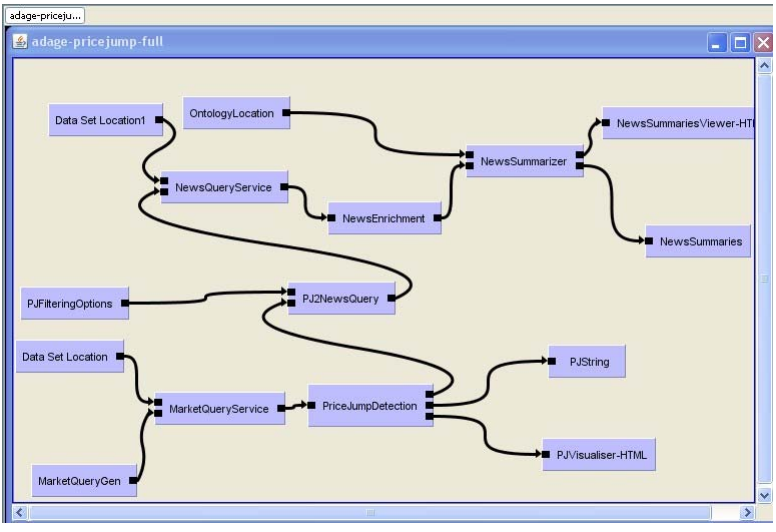

Fig. 2. Price Jump Detection Workflow with News Summarisation – showing all the services

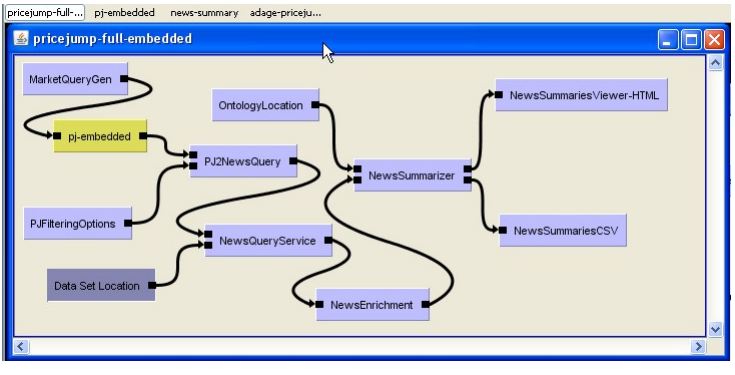

Fig. 3. Price Jump Detection Workflow with News Summarisation – with embedded components – showing a hierarchy

4.4 Specifying the Complete Process with an Embedded Component

As analysis processes can become large and complex, it is possible to abstract out certain parts of the process as embedded components. The ability to decompose complex processes into a hierarchy of components improves readability and gives researchers the ability to make changes more easily. Figure 3 shows the previous workflow with price jump workflow abstracted out as an embedded component. This technique of combining components into a single one also provides the

ability to utilize additional third party libraries which offer a similar capability – provided the interface of the component does not change.

5 Related Work

Related work falls under two categories. The first category comprises a wealth of specialised software tools for analysing financial data available both publicly and commercially. Our aim is not to provide yet another set of tools. Instead, our approach provides an opportunity to integrate several tools working together to support user-defined analysis processes. Through our prototype implementation, we demonstrated that it is possible to empower financial analysts to build complex processes through the reuse and composition of services. The prototype is also built in an *open* manner, allowing many variations and optimisations. On one hand, all components are only accessible through service interfaces meaning that any data source, software component or packaged tool can be integrated into the system. On the other hand, the definition of the Triana workflows is in XML meaning they can be converted to other representations like BPEL.

The second category of related work includes approaches for composing Web services [10], particularly those focusing on user-oriented composition tools [6,5]. The main difference with our work is that such frameworks are too general to be of practical use in such a specialised domain. Our event-based model and categorisation of services provides users with high-level domain-specific abstractions that facilitate the definition of complex analysis processes. The component composition approach used here, also shares similarities with previous work on "Problem Solving Environments" (PSEs) in computational science. In many ways, a PSE is seen as a mechanism to integrate different software construction and management tools, and application specific libraries, within a particular problem domain. One can therefore have a PSE for financial markets [2], for gas turbine engines [4], etc. Focus on implementing PSEs is based on the observation that scientists who used computational methods had to write and manage all of their own computer programs. However computational scientists must now use libraries and packages from a variety of sources, and those packages might be written in many different computer languages. Due to the wide choice of libraries available, navigating this large design space has become its own challenge. In the same way, our approach focuses on the development of an environment similar to a PSE for analysing financial data and relating this to news events.

6 Conclusion

In this paper, we described an event-based conceptual model and an environment for integrated analysis of financial market and news data. So far, this work has focused on building a proof-of-concept prototype that demonstrates the feasibility of the proposed approach. The initial case study illustrated the high level nature of analysis processes defined as workflows and demonstrated that it is possible to easily modify or adapt these workflows to handle different requirements. The

main limitation is that validation has been limited to testing the "plumbing" between the different components of the workflow. Current work-in-progress involves expanding the case study and validating the proposed prototype with finance researchers interested in finding correlations between news and financial data. As part of this effort, the event-based model will be extended and new components and services will be developed thus increasing the system's applicability and effectiveness. The long term goal is to release these components and services to the user community for public use.

Acknowledgements

The work described in this paper is a part of a large project called Ad-hoc DAta Grid Enviroments (ADAGE) which is funded by the Australian Government's DEST Innovation Science Linkage (ISL) Scheme. We are grateful to the Securities Industry Research Centre of Asia-Pacific (SIRCA) for providing the financial data used in this research. We also would like to thank Maurice Peat, Aim Mangkorntong, Rachida Ouysse, Kader Lattab, Quang-Khai Pham , Fletcher Cole and Hairong Yu for helping with the definition of the case study and the implementation of the prototype.

References

1. Back, K.: Asset Pricing for General Processes. Journal of Mathematical Economics 20, 371–395 (1999)
2. Bunnin, F.O.: Design of problem-solving environment for contingent claim valuation. In: Sakellariou, R., Keane, J.A., Gurd, J.R., Freeman, L. (eds.) Euro-Par 2001. LNCS, vol. 2150, pp. 935–938. Springer, Heidelberg (2001)
3. Barndorff-Nielsen, O.E., Shephard, N.: Econometrics of Testing for Jumps in Financial Economics Using Bipower Variation. Journal of Financial Econometrics 4(1), 1–30 (2006)
4. Fleeter, S., Houstis, E., Rice, J., Zhou, C., Catlin, A.: GasTurbnLab: A Problem Solving Environment for Simulating Gas Turbines. In: Proceedings of 16th IMACS World Congress, pp. 104–105 (2000)
5. El-Gayyar, M.M., Alda, S.J., Cremers, A.B.: Towards a User-Oriented Environment for Web Services Composition. In: Proc. Fourth Workshop on End-User Software Engineering, WEUSE IV (In conjunction with ICSE 2008), pp. 81–85 (May 2008)
6. Gavran, I., Milanovic, A., Srbljic, S.: End-User Programming Language for Service-Oriented Integration. In: Proc. of 7th Workshop on Distributed Data and Structures (WDAS 2006), CA, USA (2006)
7. EO 600 Gravitational Wave Project, `http://www.geo600.uni-hannover.de/` (last Accessed April 2008)
8. Hamilton, J.D.: Time Series Analysis. Princeton University Press, Princeton (1994)
9. Mangkorntong, P., Rabhi, F.A.: A domain-driven approach for detecting event patterns in E-markets: A case study in financial market surveillance. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 147–158. Springer, Heidelberg (2007)

10. Milanovic, N., Malek, M.: Current Solutions for Web Service Composition. IEEE Internet Computing, 51–59 (November-December 2004)
11. IPTC Web, NewsML (2008), http://www.newsml.org/ (last accessed May 5, 2008)
12. News.com ontology viewer (2008), http://infosthetics.com/archives/2005/10/cnet_newscom_ontology_viewer.html (last accessed May 2008)
13. Pham, Q.-K., Saint-Paul, R., Benatallah, B.: Time-Aware Content Summarization of Data Streams., Technical Report UNSW-CSE-TR-0722 (December 2007)
14. Sanchez-Fernandez, L., Fernandez-Garca, N., Bernardi, A., Zapf, L., Penas, A., Fuentes, M.: An experience with Semantic Web technologies in the news domain. In: Proc. 4th Int. Semantic Web Conference (ISWC 2005), Workshop Semantic Web Case Studies and Best Practices for eBusiness, Galway, Ireland (2005)
15. The Sekt Project (2008), http://www.sekt-project.com/ (last accessed May 2008)
16. Shirreff, D.: The Human Factor. Euromoney (321), 30–35 (January 1996)
17. The Securities Industry Research Centre of Asia-Pacific (SIRCA), Taqtic On-line System (2008), https://taqtic.sirca.org.au/ (last accessed August 2008)
18. Rana, O.F., Ali, A.S., Taylor, I.J.: Web Services Composition for Distributed Data Mining. In: Katz, D. (ed.) Proc. of ICPP, Workshop on Web and Grid Services for Scientific Data Analysis, Oslo, Norway, June 14 (2005)
19. Taylor, I., Shields, M., Wang, I., Philp, R.: Grid Enabling Applications using Triana. In: Proc. Workshop on Grid Applications and Programming Tools. In: Conjunction with GGF8. Organized by: GGF Applications and Testbeds Research Group (APPS-RG) and GGF User Program Development Tools Research Group, UPDT-RG (2003)
20. Taylor, I., Wang, I., Shields, M., Majithia, S.: Distributed computing with Triana on the Grid: Research Articles. Concurrency and Computation: Practice & Experience 17(9), 1197–1214 (2005)
21. Reuters, T.: Calais Web Service (2008), http://www.opencalais.com/ (last accessed August 2008)
22. The Protégé Ontology Editor and Knowledge-Base Framework (2008), http://protege.stanford.edu/ (last accessed August 2008)
23. Saint-Paul, R., Raschia, G., Mouaddib, N.: General purpose database summarization. In: Proc. International Conference on Very Large Databases (VLDB 2005), pp. 733–744 (August 2005)
24. Tsay, R.S.: Analysis of Financial Time Series: Financial Econometrics. Wiley Series in Probability and Statistics (2002)
25. Wharton Business School, Wharton Research Data Services (2008), http://wrds.wharton.upenn.edu/ (last accessed August 2008)
26. Woodroof, J.: How to Link Web Data. Journal of Accountancy, 55–58 (March 1999)
27. Zwick, S.: Not Your Father's Trading Technology. Futures, 72–75 (March 2005)