Data Mining Model Management to Support Real-time Business Intelligence in Service-Oriented Architectures

Ismail Ari[1], Jun Li[1], Alex Kozlov[2], Mohamed Dekhil[1]

[1] Hewlett-Packard Laboratories,
1501 Page Mill Rd Palo Alto, CA, USA, 94304
{Ismail.Ari, Jun.Li, Mohamed.Dekhil}@hp.com, Alex.Kozlov@turn.com

Abstract. Use of predictive models for making business-critical decisions is on the rise. However, serious challenges remain on managing data mining models and integrating them with business services using service-oriented architectures (SOA) to provide real-time Business Intelligence (BI). These challenges include model aging, management scalability, timely-communication among parties on model changes, semantic gap on interpreting models, and business process integration. We describe a data mining model management system that addresses these challenges to support sustainable and operationalized BI.

Keywords: Model Management, Business Intelligence, BI, SOA, Business Process, BPM, Business Rules.

1 Introduction

Business analysts and marketing experts in financial, telecommunication, and retail industries collect huge amounts of data on sales, customer behavior and partner profiles. They use data mining algorithms to detect patterns in historical data and to forecast future outcomes. Data mining models require deep understanding of complex statistics and algorithms as well as in-depth domain knowledge. It takes complex and composite models to predict the next purchase of customers who buy diapers, or browse digital cameras or finance offers. These models have a high initial development cost both time-wise and monetary. Therefore, we should consider models as valuable business assets and provide system support to increase their sharing and utility in the enterprise.

There are serious challenges regarding building, updating and sharing complex data mining models across the enterprise: (1) All models inevitably age over time and their predictive performance changes as the products, customers, and business environments change (especially in the Electronics domain that offers different generations rapidly). For example, the concept of a “high-end” digital camera shifts from 3 to 5 to 10 Megapixels within a few years. One has to continuously feed new

[2] Alex Kozlov was working at HP during the making of this work.

data into a model, monitor its performance and constantly tune parameters to retain good results. (2) Mining algorithms can now be easily obtained from off-the-shelf BI suites [9], [10], [12] and it is common to find practical BI deployments that incorporate hundreds of data mining models in banks, retailers, insurance companies, telcos and even casinos [8], [13]. However, there is lack of support for effectively managing and utilizing these large collections. Manual management is impractical, faulty, and unresponsive to quick change. (3) Then, there is the issue of timely-communication among the business people and the statisticians or model developers. Large-scale and dynamic nature of the models and the organizations makes it impractical to timely inform multiple parties about model-related events. Models that don't catch emerging patterns or forecast accurately are detected only after serious business consequences. Today, stakeholders email each other and developers overwrite model parameters causing the valuable interactions to be lost or buried in emails and databases. Meanwhile, the business opportunities have been lost or the customer problems have been aggravated. (4) Finally, there is a semantic gap between statisticians who talk about regressions, accuracy, confidence, and ROC vs. business analysts who talk about customer retention strategies, addressable markets, *etc.*

Overall, current data mining systems have not effectively addressed the challenges mentioned above. Specifically, we observed these problems during the development of HP Labs' Retail Store Assistant (RSA) platform [1], which aims to provide real-time personalized offers to customers through multiple retail channels including kiosks, web, and mobile devices using data mining models as well as business rules. The personalized coupons are presented to users by combining information from a user's current shopping list, user's past profile [2], business rules set in the frontline campaign process, and BI gained from this and other users' overall purchase behaviors (*e.g.* via clustering). For example, the system uses data mining models to predict customer's preferences and best matching products. Each component is published as a web service and the overall design uses a SOA.

2 Overview of Our Solution

The issues with model complexity due to business complexity cannot be simplified magically (see "No Magic" in [8]). Tools can help sift through large data and help store models, but it takes coordination among people in different roles, different departments, and even different organizations (due to partnership or outsourcing) to discover actionable knowledge. Yet, we can design systems to increase access of multiple stakeholders to model development process, automate tedious and repetitive tasks, reduce delays, and finally capture rich information to track model provenance and help with future inquires and management.

We developed a system solution that can track a data mining model lifecycle from creation, to business inclusion, to performance deterioration ("aging" or "decay"), to maintenance, and finally to expiration and archival. It also tracks rule-model and model-to-model dependencies, so that if a model needs to be evicted from repository due to poor performance, then related entities that depend on this model are informed promptly. We track dependencies over multiple repositories through a workflow in a

business process management (BPM) system with web service capabilities described below. We also define rich and extensible set of model-related metadata to track model provenance and capture the collective intelligence that is today left in experts' minds and in silos of proprietary tools. Rich metadata helps close the semantic gap between business analysts and model developers leading to actionable BI. For example, in our retail kiosk scenario two data mining models for customer *behavior change detection* and *fuzzy product matching* used by the campaign process would be maintained by the model management system described here.

2.1 Model Lifecycle Management

We implemented a special workflow to orchestrate data mining model management, specifically to handle model lifecycle-related events illustrated in Figure 1, namely model created, model referenced and unreferenced (by a business process or rule), model deteriorated, model to be expired, and expired states and the triggering events. Today, a BPM system lies at the core of many SOA offerings providing the language and tools to express business logic in an executable form (*e.g.* BPEL [15]). We used Microsoft Biztalk BPM, but any other system allowing visual or programmatic description of workflow logic could have been used.

Figure 2 shows a partial diagram of the orchestration that synchronizes the communication of business analysts, statisticians and the automated model performance evaluation routines over model-related events. The orchestration publishes a receive port as a web service to receive model-related event calls and queries (top-left in Fig.2). Note that this service is for management purposes and not for applications to query a specific model's prediction. Internally, the workflow calls model metadata repository to store or query model metadata and to advance models' status. Details of repository Application Programming Interface (API) for adding-removing models, getting model tags/authors, adding dependencies, *etc.* are straightforward and therefore skipped for brevity. Model repository is currently implemented as a .NET library and the data is stored in Microsoft SQL Server.

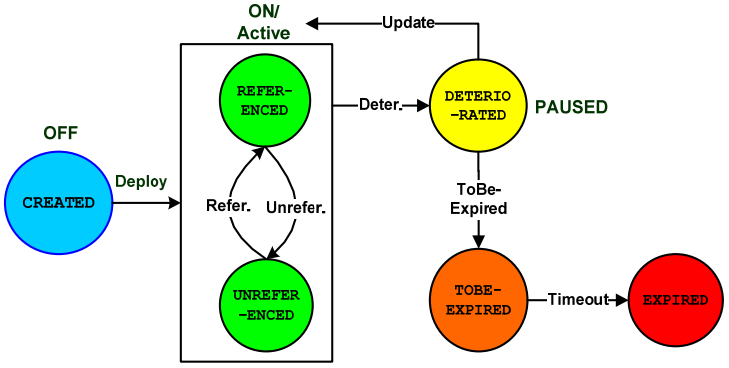


Fig. 1. Data mining model lifecycle is tracked by a hierarchical state machine. The unique feature of this lifecycle is the inclusion and tracking of business-level dependencies.

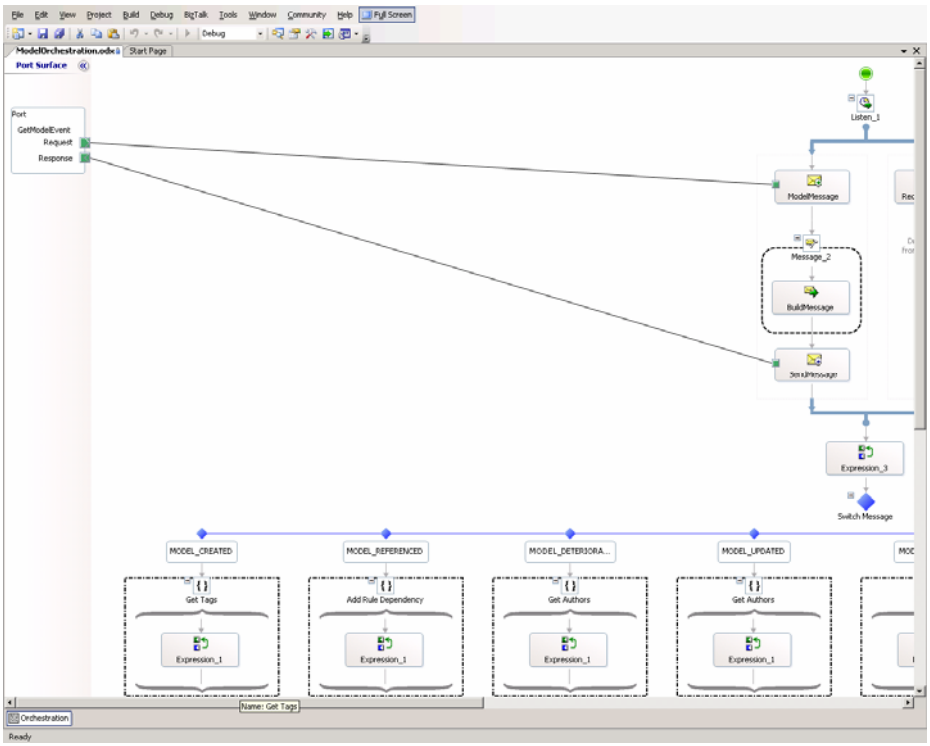


Fig. 2. A screenshot from the data mining model management workflow implemented in Microsoft Biztalk BPM system.

The orchestration first checks the model event type in the event payload and switches to the branch associated with that type of event that represents different model states shown in Figure 1. Model is assumed to be in the creation state until it is deployed. A MODEL CREATED (or deployed) event informs the orchestration that statisticians have deployed a new model into the model repository. Statisticians can raise this event by calling the web service published by the orchestration. The event carries the model identifier and other basic model attributes (described later). The new model goes to an Active state and starts to get managed by our system. Note that the state machine in Figure 1 is hierarchical where the Active state embeds REFERENCED and UNREFERENCED sub-states. Models that enter Active or ON state are initialized to Unreferenced sub-state. The Active state also keeps the history of the last state (for each model), so that it can return to this state when models exit and reenter Active state. It is also possible to view the state machine as consisting of On, Off, and Paused states.

The related business analysts and other users will be informed when a new model is created and deployed. The orchestration does this by compiling a list of subscribers based on subscription keywords and notifying them. If analysts choose to integrate this model's prediction in their business rules or processes, then they reference the model's endpoint (*e.g.* Web Service URI recorded in metadata) and raise the MODEL

REFERENCED event to inform and be informed about future model status updates. Processes can reference model services directly or via a rule-model binding. The Reference event payload includes a unique business rule id and the model id and the orchestration adds the rule-model dependency information to the given model's metadata in the model repository. When model developers receive MODEL DETERIORATED events as a result of the orchestration's periodic performance scans (described later) they determine if they can fix the model by updating internal parameters and if so they raise a MODEL UPDATED event. The orchestration handles update notifications similar to deteriorations by forwarding them to related people such as authors of this model and dependent business rules. If the model cannot be fixed (*e.g.* due to products becoming obsolete), then developers can choose to set a model expiration time and raise a MODEL TOBE-EXPIRED event. The orchestration will find the business rules dependent on this model using the model repository API and notify the associated rule authors. It also sets a timer, so that a MODEL EXPIRED event is raised at the expiration time. When the rule authors receive expiration notification, they revise their business rules based on other active models in the repository, remove references to expiring models, and raise the MODEL UNREFERENCED event to the orchestration. Note that this is a viable support extension to business process exception handling as the failures are detected and fixed at the background when they occur and not when there is a customer request with real-time expectations. When a model finally expires, business rules won't be able to access it to get predictions and the orchestration removes the business rule ids from the model metadata. If no pending references exist, the orchestration directly raises a MODEL EXPIRED event to finish the process. This will remove the model from the system and no further notifications related to this model will be raised.
Note that data mining model lifecycle management and metadata collection (described next) is representative of the Business Service Management (BSM) functionalities such as configuration, measurement, fault and trouble ticketing, and inventory management proposed as extensions to the best practice frameworks in IT service management such as the IT Infrastructure Library (ITIL) v3 [6].

2.2 Metadata for Data Mining Models

Figure 3 shows basic model attributes including the globally unique identifier (GUID), model name that provides a quick verbal reference, and the textual description that explains what the data mining model is about. We also track model creation, last access, and expiration times. Author field lists the preferred contacts of developers that need to be informed about model lifecycle events. Data mining algorithm specifies particular algorithms (decision tree, logistic regression, clustering, naive Bayesian, neural network, *etc.*) used in construction of this model. Other metadata fields include model schema for describing I/O attributes, model assumptions, tags or keywords, training dataset information, performance evaluation methods, event triggers and thresholds values, rules that depend on this model, and finally inter-model dependencies.

Fig. 3. Data mining model metadata and the details of basic attributes.

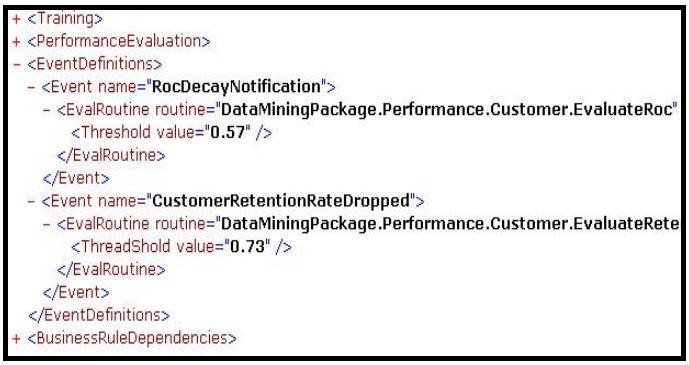


Fig. 4. Model performance evaluation and event triggering parameters ("EventDefinitions") are shown. They describe which function to call when a threshold is passed.

An input attribute under SchemaDefinitions (not shown due to space) can be selected simply from a column in the database (*e.g.* CustomerId) or it can be an aggregated attribute (*e.g.* The total purchases over the last 3 months). The output attribute represents the result/attribute that the data mining model is trying to calculate or predict (*e.g.* The top 10 coupons to offer, churn rate, retention rate, or customer response probability). In addition, a data mining model works best (or only works) under certain conditions. The model developer can document these model assumptions (*e.g.* data ranges, input data quality, *etc.*) during the model construction and update them with gained knowledge over time. This rich information, beyond simple versioning and dependency tracking, gets transferred to the peer model

builders or business analysts who rely on the correct operation of models when making business decisions or creating business rules. Our system will potentially reduce the number of failures or exceptions due to violation of undocumented assumptions. Image a practical business intelligence system containing hundreds to thousands of different models; it would be a daunting task to identify these buried assumptions, even if they could eventually be discovered by inspecting the entire data mining model. As model complexity increases and inter-model and model-rule dependencies proliferate, a model management system such as ours becomes a necessity. An analyst or manager cannot sift through raw database tables, try to understand SQL queries, or even locate the models during a business chaos such as all "personalized" coupons coming out the same, customers refusing to pay, out-of-stocks occurring, *etc.*

Models can be tagged by their authors to allow indexing and textual search. By querying a tag, we retrieve models that share the same or similar tags, thus finding models that are semantically linked or related to each other. This helps with model selection process before model composition. More interestingly, social tagging can be applied through our management system creating a perfect application for Enterprise 2.0 (*i.e.* Web 2.0 for the enterprise).

Business rules are also assumed to be represented by GUID in their respective repository (*e.g.* as in Microsoft Business Rule Engine). When a model-change event happens (*e.g.* model deterioration) the author of a dependent business rule gets notified by the orchestration. Different models can also depend on each other through versioning or other taxonomies. For example, a model can be the result of back-fitting of another model, thus having an "improved model" (parent-child) relationship. Models can also share the same data source, but might use different data mining algorithms, thus having a "Models Sharing Training Dataset X" (peer or siblings) relationship. Similarly models can be composed using machine learning to create a supermodel and this would create both parent-child and sibling relationships. Our system, through the use of rich metadata, allows tracking and managing different types of complex model relations. A tool that can crawl through the data mining model repository and highlight these relations visually would be extremely valuable. However, we do not focus on this topic in our research as there are several products that can enable us to add this capability.

Developers can use their favorite BI tools to build data mining models. Next, they can export their model schema (*e.g.* using the Predictive Model Markup Language-PMML [11] format) and enrich that with metadata such as our basic model attributes, model assumptions while also exposing their performance evaluation routines through public API (SOAP, HTTP, message queues, *etc.*) to be called by our orchestration periodically or per-event-based. Associated with the performance evaluation routines, they define event trigger predicates shown in Figure 4. Simple examples include: "ROC < 0.57" or "customer retention rate < 0.73". The orchestration scans through the associated performance evaluation results and raises the related event (*e.g.* MODEL DETORIATED) if the criteria evaluates as TRUE.

3 Prototype Implementation

We built the orchestration in Figure 2 and the model metadata repository. We demonstrated the lifecycle tracking of a few models as shown in Figure 5, which shows simulated model events for our CouponPrediction and InkUsage models. The date information denotes that coupon model was created, referenced, deteriorated and updated in a day time and then referenced again the next day. We can expect models to evolve much slower in real-life. However, the detection and handling of critical model events has to be done in real-time in order to assure healthy operation of systems depending on the BI gained from these models. Figure 6 shows ROC-based model performance comparison of two CouponPrediction data mining models for two categories. It highlights models with deteriorated performances with a red color. We are currently in process of evaluating our system on performance and scalability.

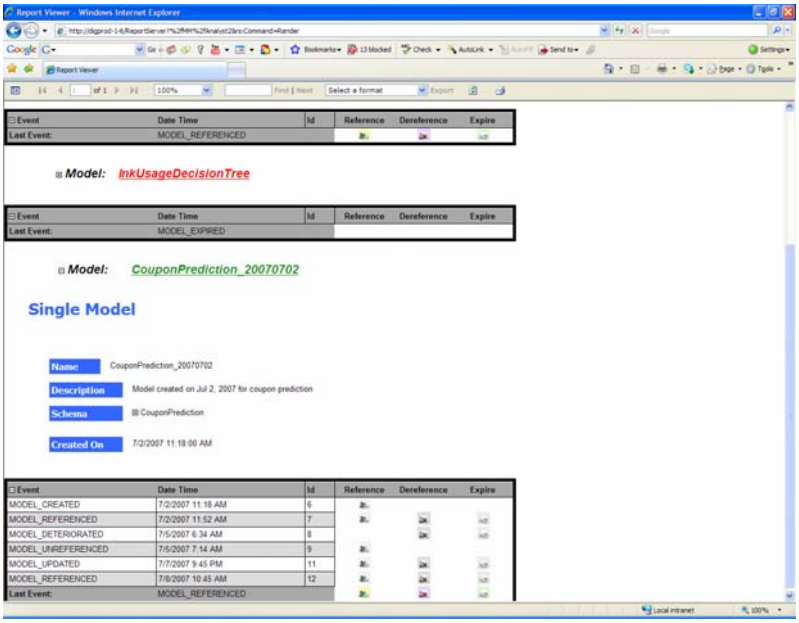
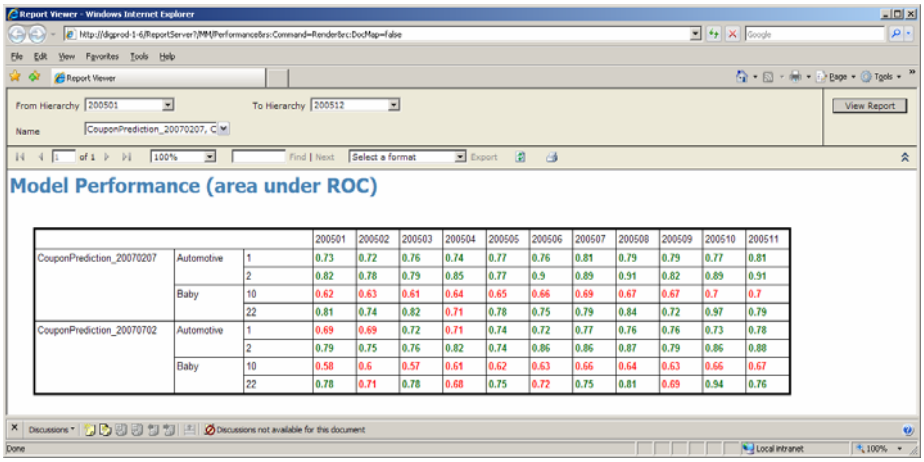


Fig. 5. Web front-end to model management system showing models in our repository and their status.

4 Related Work

We address challenges that start after models are built. Our goal is to let developers use their favorite tools (Microsoft, SAS, FairIsaac, Oracle) to build data mining models and then register them into our model management system. In other words, this work does not focus on the details of algorithmic issues and attribute/parameter selection in model construction, which differentiates our work from most existing tools and systems [9], [10], [12], [3]. No prior art addresses integration of data mining



			200501	200502	200503	200504	200505	200506	200507	200508	200509	200510	200511
CouponPrediction_20070207	Automotive	1	0.73	0.72	0.76	0.74	0.77	0.76	0.81	0.79	0.79	0.77	0.81
		2	0.82	0.78	0.79	0.85	0.77	0.9	0.89	0.91	0.82	0.89	0.91
	Baby	10	0.62	0.63	0.61	0.64	0.65	0.66	0.69	0.67	0.67	0.7	0.7
		22	0.81	0.74	0.82	0.71	0.78	0.75	0.79	0.84	0.72	0.97	0.79
CouponPrediction_20070702	Automotive	1	0.69	0.69	0.72	0.71	0.74	0.72	0.77	0.76	0.76	0.73	0.78
		2	0.79	0.75	0.76	0.82	0.74	0.86	0.86	0.87	0.79	0.86	0.88
	Baby	10	0.58	0.6	0.57	0.61	0.62	0.63	0.66	0.64	0.63	0.66	0.67
		22	0.78	0.71	0.78	0.68	0.75	0.72	0.75	0.81	0.69	0.94	0.76

Fig. 6. Model performance comparison where deteriorated models are marked with red color.

model collections with business processes to automatically provide business insights, while also addressing the model aging, scalability, timely-communication and semantic gap challenges. Microsoft Analysis Services (MSAS) provides Analysis Management Objects (AMO) library to create, modify and delete data mining objects such as cubes and mining models. Their "collections" can contain only mining models built on the same data whereas our system can cover and relate all mining models in an enterprise. Furthermore, we use model-dependency tracking to relate models to rules and other models. Dependency tracking and decay monitoring capabilities of SAS Enterprise Miner, FairIsaac Model Builder, and Oracle Data Mining & Analytics are limited compared to our system. Data mining models are more complex constructs then web service descriptions (WSDL) and API definitions. Therefore, service and application lifecycle management systems [5] focusing primarily on versioning do not solve problems addressed here.

Recent research has shown the importance of making models first-class citizens of database and called these special repositories modelbases [7]. Yet, there hasn't been any work providing integration of data mining modelbases with business processes to automatically provide business insights, while also addressing the model aging, scalability, timely-communication and semantic gap challenges mentioned before. There also has been increased interest in real-time ETL [14] and data stream processing recently. Our work is remotely related to but significantly different from the previous research on generic model management. For example, in their "model management 2.0" paper Bernstein and Melnik describe techniques for schema mapping and matching and propose a system to track the evolution of the model schema and mappings among them. These topics are complementary to our current system in providing real-time BI. We recently also addressed stream and complex event processing issues in another paper [16].

5 Conclusions

Serious challenges on managing data mining models and integrating them with online business services have been addressed in this paper. The addressed challenges include rapid handling of model deterioration, timely-communication among parties, closing the semantic gaps on model interpretations via rich metadata, business process integration and management scalability. We designed and implemented a data mining model management system and tested it with a few data mining models related to retailing. We accessed and used data mining model predictions via web services from our retail platform. Our goal is to provide a sustainable, real-time BI support for online business services and increase model utility and value across the enterprise. We are working on extending our work with new management features and testing system scalability with larger model collections. We plan to continue our work by running usability tests to tackle presentation issues emerging with large-scale deployments. These ethnographic studies will be major contributions towards delivering personal analytics or "BI to the masses". Along the same direction we're investigating ways to provide model management as a service in the cloud.

Acknowledgments. We'd like to thank Henry Sang and Meichun Hsu for supporting this collaborative work and anonymous reviewers for their helpful comments.

References

1. Ari I., Li J., Ghosh R., Dekhil M., Providing Session Management As a Core Business Service, WWW 2007.
2. Bernstein P.A. and Melnik S, Model Management 2.0: Manipulating Richer Mappings, SIGMOD, 2007.
3. FairIsaac Enterprise Decision Management (EDM) http://www.fairisaac.com
4. Ghosh R. and Dekhil M., Mashups for Semantic User Profiles, WWW, April 2008.
5. HP SOA Systinet Registry/Repository, http://www.hp.com/go/soa
6. IT Infrastructure Library (ITIL) v3, http://itil.org
7. Liu B., Tuzhilin A., Managing large collections of data mining models, Communications of the ACM, Vol 51. No:2, Feb 2008
8. Michael Meltzer, Using data mining on the road to successful BI, Parts II-III, DMReview.com, Sep-Oct 2004.
9. Microsoft SQL Server 2005 Analysis Services, http://www.microsoft.com/sql/technologies/analysis/
10. Oracle Data Mining, http://www.oracle.com/technology/products/bi/odm
11. PMML, Predictive Model Markup Language, http://www.dmg.org/pmml-v3-2.html
12. SAS Model Manager, http://www.sas.com/technologies/analytics/modelmanager/
13. Sumathi S. and Sivanandam S. N., Data mining in customer value and customer relationship management, Studies in Computational Intelligence 29, 321-386, 2006. Springer-Verlag.
14. Thomsen C, Pedersen T., Lehner W., RiTE: Providing On-Demand Data for Right-Time Data Warehousing, ICDE 2008
15. Web Services Business Process Execution Language (WSBPEL), OASIS, http://www.oasis-open.org
16. Wei M, Ari I., Li J., Dekhil M., ReCEPtor: Sensing complex events in data streams for SOA, HPL-2007-176, 2007.