MASTERING INTELLIGENT CLOUDS
Engineering Intelligent Data Processing Services in the Cloud

Sergiy Nikitin, Vagan Terziyan, Michal Nagy
Industrial Ontologies Group, University of Jyväskylä, Mattilanniemi 1, Jyväskylä, Finland
sergiy.nikitin@jyu.fi, vagan.terziyan@jyu.fi, minagy@jyu.fi

Keywords: Agent Technology, Cloud Computing, Semantic Web, Cloud Services, Ubiware.

Abstract: Current Cloud Computing stack mainly targets three architectural layers: Infrastructure, Platform and Software. These can be considered as services for the respective layers above. The infrastructure layer is provided as a service for the platform layer and the platform layer is, in turn, a service for the Software layer. Agent platforms fit the "Platform as a service" layer within this stack. At the same time, innovative agent-oriented approaches to programming, open new possibilities for software design in the cloud. We introduce main characteristics of our pilot agent platform called UBIWARE and offer flexible servicing architecture within the cloud platform, where various components and systems can configure, run and reuse intelligent cloud services to provide higher degree of flexibility and interoperability for their applications.

1 INTRODUCTION

Fast development of network technologies has recently brought back to life business models with the "thin client" architecture. Powerful data centers connected to the internet via broadband networks can minimize IT-infrastructure of any company to a set of simple terminals with less demanding system requirements. All the software and data can reside on the data center side, making user access easy and location independent. The providers offer different payment schemes as "pay-per-use" or subscription-based, that seems to be advantageous compared to standard IT-infrastructure expenses. The approach has got a set of new features and a new branding name: "Cloud Computing" (Hayes, 2008; Foster, 2008).

Cloud management platforms provide API for management either through command line or a remote method calls. The API, however, is used mainly by system administrators, who take care of proper functioning of services within the cloud. Management of the cloud platform is considered as something that a system administrator should arrange and do. Usually administrators use batch files for managing routine tasks and resolving exceptional situations.

At the same time, more and more software architecturing paradigms call for new approaches to software design and development, where software components get a certain degree of self-awareness, when a component can sense its own state and act based on the state changes. The vision of Autonomic Computing (Kephart, 2003) proposes to handle the complexity of information systems by introducing self-manageable components, able to "run themselves." The authors state, that self-aware components would decrease the overall complexity of large systems. The development of those may become a "nightmare of ubiquitous computing" due to a drastic growth of data volumes in information systems as well as heterogeneity of ubiquitous components, standards, data formats, etc. The Cloud Computing and Autonomic Computing paradigms will become complementary parts of global-scale information systems in the nearest future. Such a fusion sets the highest demands to the software architects because cloud platforms will have to provide self-management infrastructure for a variety of complex systems residing in the same cloud, separated virtually, but run physically on the same hardware. At the same time, the cloud platform itself

may possess features of self-aware complex system. A variety of self-aware components of different nature (i.e. end-user oriented, infrastructure-oriented, etc.) will need a common mechanism for interoperability, as far as they may provide services to each other.

The vision of GUN – Global Understanding Environment (Terziyan, 2003, 2005; Kaykova et al., 2005) has introduced a concept of "Smart Resource" and a notion of an environment where all resources can communicate and interact regardless of their nature. In GUN various resources can be linked to the Semantic Web-based environment via adapters (or interfaces), which include (if necessary) sensors with digital output, data structuring (e.g. XML) and semantic adapter components (e.g. XML to Semantic Web). Software agents are assigned to each resource and are assumed to be able to monitor data coming from the adapter about the state of the resource, make decisions on behalf of the resource, and to discover, request and utilize external help if needed. Agent technologies within GUN allow mobility of service components between various platforms, decentralized service discovery, utilization of FIPA communication protocols, and multi-agent integration/composition of services.

When applying the GUN vision, each traditional system component becomes an agent-driven "smart resource", i.e. proactive and self-managing. This can also be recursive. For example, an interface of a system component can become a smart resource itself, i.e. it can have its own responsible agent, semantically adapted sensors and actuators, history, commitments with other resources, and self-monitoring, self-diagnostics and self-maintenance activities.

In this paper we introduce a flexible servicing architecture within the cloud platform, where various components and systems can configure, run and reuse intelligent cloud services to provide higher degree of flexibility and interoperability for their applications. We use our pilot agent platform developed in accordance with GUN vision called UBIWARE to show how cloud computing can expand platform functionality and at the same time how an agent platform can become a high-level service provisioning instrument in the cloud.

The paper is organized as follows: In the next Section we discuss architectures of state-of-the-art cloud computing platforms and explore the possibilities for the interoperability mechanisms. Section 3 presents the architecture of the semantic middleware agent platform and explores possible options of connectivity with the cloud. Section 4 describes the scenarios and the architecture of the agent-driven intelligent servicing platform for a cloud. In Section 5 we discuss related work and conclude in Section 6.

2 STATE OF THE ART IN CLOUD INTELLIGENCE PLATFORMS

Architecture of current Cloud Computing stack mainly targets three layers: Infrastructure, Platform and Software. These layers can be considered as services for the respective layers above. The infrastructure as a service (IaaS) is provided to the platform layer and the platform becomes a service (PaaS) for the software layer. And finally, the software as a service layer (SaaS) brings the topmost end-user web services to clients (see Figure 1).

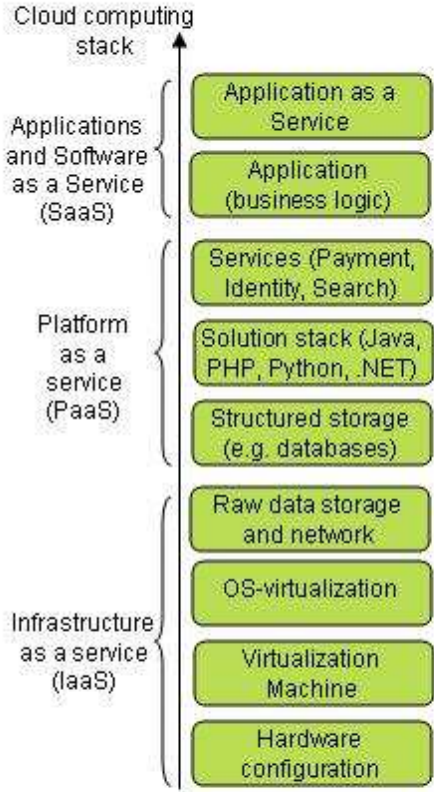

Figure 1: Cloud computing stack

Cloud providers market niche is already a competitive field. Several big players are currently active in the market, e.g. SalesForce.com (SFDC), NetSuite, Oracle, IBM, Microsoft, Amazon EC2, Google Apps, etc. For a comprehensive survey of cloud computing systems see (Rimal et al., 2009). The services of the platform layer are in the scope of

this work. In the next section we present a middleware platform and later introduce a new servicing approach in the cloud stack.

3 UBIWARE PLATFORM

UBIWARE has two main elements: an agent engine, and S-APL – a Semantic Agent Programming Language (Katasonov and Terziyan, 2008) for programming of software agents within the platform. The architecture of UBIWARE agent (Figure 2) consists of a *Live* behavior engine implemented in Java, a declarative middle layer, and a set of Java components – *Reusable Atomic Behaviors* (RABs).

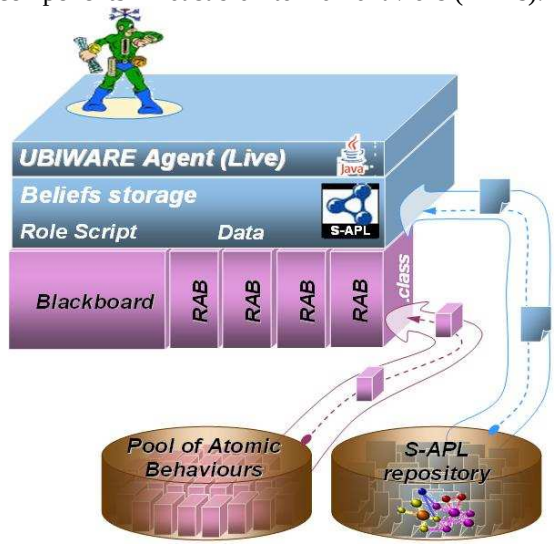

Figure 2: UBIWARE Agent.

RABs can be considered as sensors and actuators, i.e. components sensing or affecting the agent's environment, but are not restricted to these. A RAB can also be a reasoner (data processor) if some of the logic needed is not efficient or possible to realize with the S-APL means, or if one wants to enable an agent to do some other kind of reasoning beyond the rule-based one. UBIWARE agent architecture implies that a particular UBIWARE-based software application will consist of a set of S-APL documents (data and behavior models) and a set of specific atomic behaviors needed for this particular application. Since reusability is an important UBIWARE concern, it is reasonable that the UBIWARE platform provides some of those ready-made.

Therefore, logically the UBIWARE platform, consists of the following three elements:

- The Live behavior engine
- A set of "standard" S-APL models
- A set of "standard" RABs

The extensions to the platform are exactly some sets of such "standard" S-APL models and RABs that can be used by the developers to embed into their applications certain UBIWARE features.

As Figure 2 shows, an S-APL agent can obtain the needed data and rules not only from local or network documents, but also through querying S-APL repositories. Such a repository, for example, can be maintained by some organization and include prescriptions (lists of duties) corresponding to the organizational roles that the agents are supposed to play.

Technically, the implementation is built on top of the JADE – Java Agent Development Framework (Bellifemine et al. 2007), which is a Java implementation of IEEE FIPA specifications.

4 MASTERING INTELLIGENT CLOUD PLATFORM

Cloud computing providers offer various stack configurations with different sets of software and services inside. Theoretically, one can buy any configuration from the cloud provider; however this configuration will have nothing to do with the already running business logic of the customer. The application scenarios a customer wants to run will have to be adjusted. For example, consider a case, when a customer buys a virtual server with the MySQL database installed and a Java solution stack available. On top of this stack customer runs a workflow engine, e.g. BPEL-based. The user will have to install the engine, and then adjust local data storage settings (passwords, tables, queries). Then the process descriptions (BPEL files) should be adjusted to work with local settings. In some cases this process may be avoided if the cloud stack is identical to the customer's environment, and if the all code was developed as portable. But what if the cloud stack slightly differs, but the prices are very attractive? Then customers may need to spend resources for solution code porting.

We propose architecture for a generic stack extension that allows users and platform providers to:

- Smoothly integrate with the infrastructure
- Build stack-independent solutions
- Automate reconfiguration of the solutions

The architecture is based on the UBIWARE platform architecture and extends cloud platform

services with the standardized configurable intelligent models.

4.1 Agent-driven Servicing in the Cloud

Interoperability is stated as one of the challenges of the cloud computing paradigm. We believe that adoption of the existing interoperability tools and solutions will become one of the major cloud computing research directions. Dummy platform API extension will just put the interoperability problem from the cloud provider to the client side. At the same time, the competitiveness of the cloud providers may depend on the simplicity of the integration with the client solutions and systems. Therefore, we foresee that embedded services offered by the cloud providers should be flexible and smart enough to handle client-specific model adjustments and configurations. We expand the understanding of the platform service to the smart proactive agent driven service. Such a service should not only be flexible and configurable in accordance with the customers' needs, it should also be prepared to resolve data- and API-level interoperability issues while being integrated with the client software.

Figure 3 shows the placement of the agent-driven extension in the cloud computing stack. From the user perspective the extension is still a service API but it offers an advanced functionality.

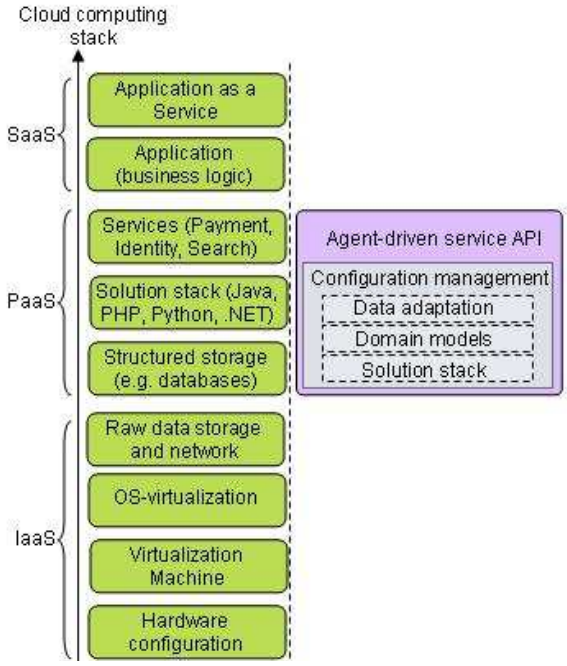

Figure 3: Placement of the agent-driven API

The API shown above is a standalone middleware platform running either as a cloud facility, or embedded into the virtual machine instances as a platform extension. The detailed API content is shown of Figure 4.

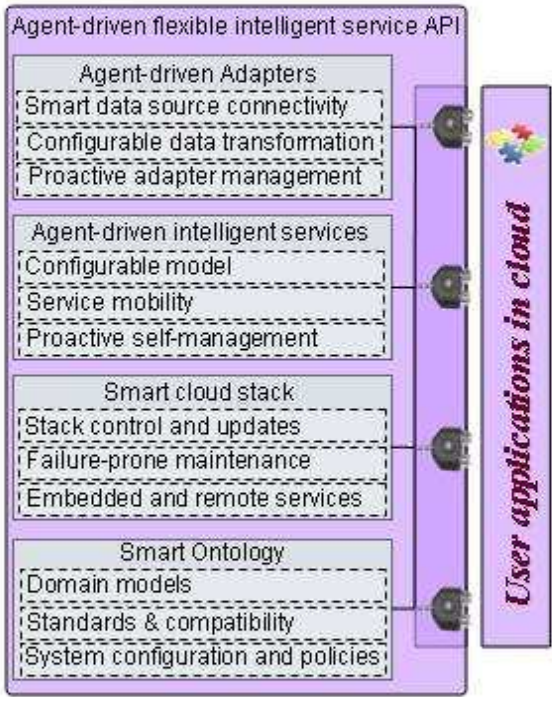

Figure 4: Agent-driven flexible platform service extension API

The agent-driven adapters are software entities that facilitate data sources management. Adapters provide advanced data source connectivity functions (e.g. simplified database connectivity, file formats parsing, sensor data acquisition, etc.). Next, adapters handle the connectivity problem by providing the components for data transformation with configurable mapping functionality. The adapter becomes a proactive entity, i.e. it observes its state and takes actions based on the state and environment changes. The actions of the adapter may vary from simple fault messaging up to self-reconfiguration when an exceptional or fault situation occurs.

The services' API allows the user to run declarative models as services. The API provides a "model player API" for a particular domain-specific model definition language (the example of the API as well as the language will be discussed in the next Section). The model of the service being played is at the same time controlled by the dedicated agent that takes care of proper model functioning (e.g. load balancing and failures in the operation). Service agent may temporarily relocate the service

executable code to another virtual machine instance to improve the performance in critical cases, thus the service becomes remote for its own original virtual machine instance. We also consider service API that has a local representative agent on each virtual machine, but the service execution is handled by the cloud provider (see Figure 5).

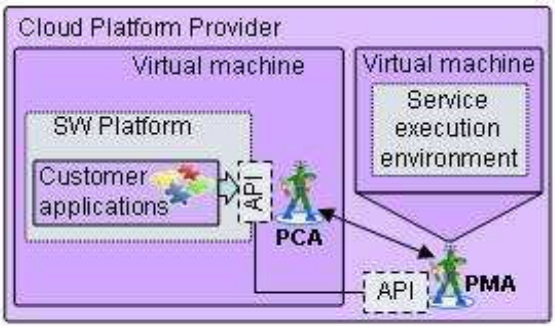

Figure 5: Service execution in cloud infrastructure

In the Figure above the PCA stands for the Personal Customer Agent and PMA is a Platform Management Agent. The PCA may request the PMA agent to host a service execution (time period is a subject of contracting details) on a separate virtual machine to obtain e.g. higher performance, or for any other reason. At the same time the local API within the user's virtual machine and/or platform will stay the same. The PCA agent will wrap and forward local API calls to the PMA agent. The difference of the architecture proposed from the standard remote method invocation is a control channel between agents that allows the service management layer to stay separated from the service consumption (service calls).

In the next Section we discuss how the web services from the data mining domain can be integrated into the infrastructure described above. Data mining services can be embedded as platform services into the cloud stack for particular domain-specific cloud configurations at the same time preserving features of configurability, mobility and self-awareness.

4.2 Mastering Data Mining Services into the Cloud

To model the data mining services we have to define a corresponding data mining domain ontology. The ontology will cover data mining methods as well as requirements for method inputs and respective outputs. The inputs and outputs should, in turn, refer to the data types. The data mining domain can not include all possible applications of its methods; therefore we should keep the granularity of the conceptualization and distinguish the data mining models and their application scenarios. For the purpose of this scenario we take two data mining techniques: cluster analysis and k-NN method (classification).

The efforts towards standardization of data mining techniques, methods and formats have been a matter of discussion for the last ten years. One of the notable efforts is PMML language (PMML, 2009; Guazzelli et al., 2009). The language is a standard for XML documents which express instances of analytical models. In our work we take PMML as a reference model for the Data Mining Ontology and enhance both the model as well as the data with the semantic descriptions required to automate data mining methods application to the domain data. In this work we do not take into account the stage of information collection, preparation, etc. We assume that data is ready for data mining algorithm application. The PMML structure for model definition is composed of a set of elements that describe input data; model and outputs (Figure 6).

Figure 6: PMML model structure

The PMML specification ver. 4.0 provides means for exhaustive model description, thus the model can be fully exported or imported without losses. Such model transportability gives huge opportunities for service orientation of the data mining methods. We

also expect the PMML models reuse in the cloud computing domain in the nearest future. The specification of a software-independent descriptive data mining standard implies that Infrastructure and Platform layers of Cloud Computing stack are fully transparent for the data mining methods, i.e. the functional characteristics of the method-based services will be same for any stack configuration. The QoS, however, may vary depending on the performance of the hardware and efficiency of platform software, therefore the additional control channel over the service configuration may be needed.

We have identified three main types of data mining services regardless of their application domain and have introduced a classification of them (Figure 7).

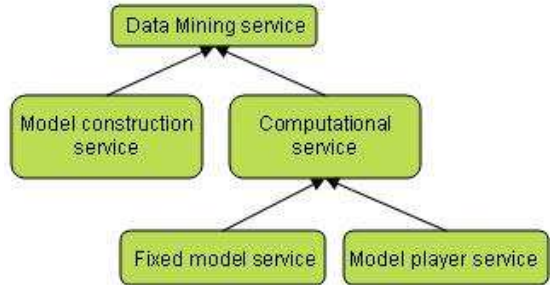

Figure 7: – Upper ontology of data mining services

We consider two major categories of data mining services:

- model construction services
- computational services

The model construction services produce a model (a semantic description) from the set of learning samples. In other words, input of such a service is learning data and conditions (for the neural network depending on its mode it can be a set of vectors plus e.g. initial network parameters). The output of the model construction service is a model with the parameters assigned (e.g. a neural network model, see Figure 8).

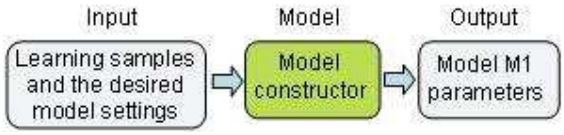

Figure 8: Model construction service

The group of computational services can also be divided into two major groups:

- services with fixed model
- model player services

The services with fixed model define the format of the input and output as well as provide reference model description and the parameters that determine how the model is configured. For example, Figure 9 shows the definition of the neural network-based alarm classifier service for a paper machine.

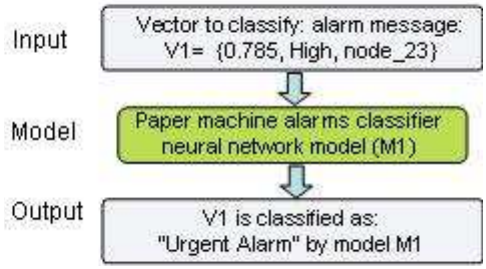

Figure 9: Neural network model in a classification service

Usage of model player services has two stages: in the first stage the service accepts the service model as an input, and in the second stage, it can serve as a fixed model service (see Figure 10).

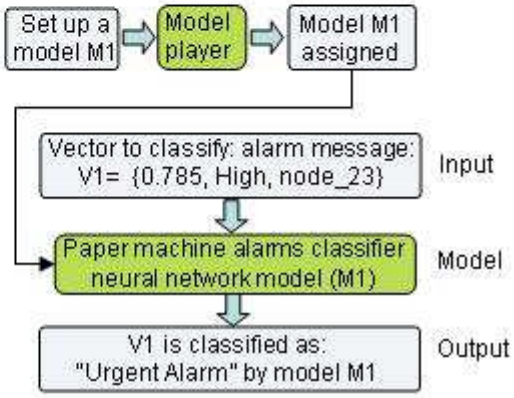

Figure 10: Model player service

The true power of data mining services can be demonstrated in combination with the distributed querying (i.e. collecting learning or classification data), data mining model construction and further classification. The generic use case of such combination is shown on Figure 11.

The automated data collection process (first step in the use case) uses the Ontonuts technology and approach researched in the (Nikitin et al., 2009). The approach allows dynamic distributed query planning and execution, which we apply in this work to collect learning set data. The sources of the data may vary from databases, to csv-files and reside physically on different platforms and sites. The data collection and, hence, the querying implemented as a sequence of semantic data service calls orchestrated by a workflow management agent. Service

orientation of data sources makes distributed querying a homogeneous part of other service-based workflow scenarios. The data collected (usually in form of a table of query results), may undergo preparatory steps to become a learning dataset. In this work we omit the procedure of normalization, or other data transformations, however, they will be necessary and obligatory. We assume that all operations with the data are also wrapped as semantic services.

As soon as the learning set is ready, a desired model constructor should be chosen (step 2). The model constructor may require specific data preparation, therefore it is good to combine data preparation step with the model constructor service.

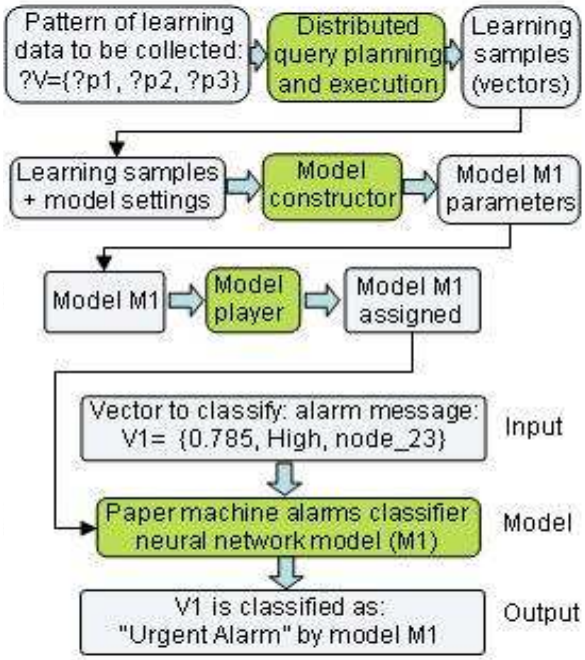

Figure 11: A use case scenario

As an input, model constructor may require additional input parameters for model building. Those may be set as default, or, if other parameters were prepared, they should be supplied in the proper form. When a model is ready, we feed it to the model player service which is a platform service in terms of cloud computing, because it provides an infrastructure and software platform for service execution. As soon as our newly built model is deployed as a service, we can start classifying the data vectors, e.g. alarms coming from the paper machine.

The scenario described above may be dynamically reconfigured by the infrastructure agents. Some steps of the case (e.g. learning) may be temporarily moved to the separate execution environment (separate platform or virtual machine) to perform computationally expensive tasks.

The overall infrastructure of services should be highly proactive and responsive to the customer calls. Agents may monitor the execution and be ready to reconfigure their services in accordance with the current customer context.

5 RELATED WORK

The cloud platform solutions for business intelligence are gaining popularity. For example, SalesForce.com claims about 2 million success stories of its customers. The platform provides wide range of products (from infrastructure as a service, up to tailored web-based solutions for health care, retail and sales). The business intelligence tools are offered too. Nevertheless, the user has to adjust or prepare her/his software and data for the tools provided by the cloud. The advantage of the approach we offer is to empower any cloud platform with the intelligent adaptation mechanisms that would allow seamless data connectivity and integration. The architecture we offer is an extension to the cloud platform, not the platform itself. The data mining services with highly configurable parameters driven by the intelligent agents would simplify business intelligence integration, hence making adoption of cloud architecture easier for clients.

6 CONCLUSIONS

The research presented above describes specific application domain of intelligent services. We foresee that model player services will be a successful business case for the emerging paradigm of cloud computing. Pay-per-use principles combined with high computational capacities of cloud and standardized DM-models will be definitely an alternative to expensive business intelligence and statistics toolkits.

Another niche of data mining services in cloud computing can be model construction services. Such systems will drive innovations in data mining methods as well as applied data mining in certain domains. Such services will compete by introducing know-how and innovative tools and algorithms that bring add-values in e.g. predictive diagnostics or

computational error estimation. This direction will lead to so-called “web intelligence” (Cercone et al.).

The role of UBIWARE platform in cloud computing emerges as a cross-cutting management and configuration glue for interoperability of future intelligent cloud services and client applications.

The main burden of UBIWARE will be management of consistency across different domain conceptualizations (Ontologies) and cross-domain middleware components. Fine-grained ontology modeling is still a challenge for research community and we predict that in the nearest future the domain modeling will be task-driven, i.e. the domain model engineers may incorporate some standardized and accepted conceptualizations, whereas the whole ontology for solution will be tailor made. Tailored ontologies will require subsequent mapping mechanisms and additional efforts.

The advanced data integration mechanisms embedded into the cloud platform as services is also an interesting concept that may become an add value for competing cloud platforms. The easiness of integration into the cloud infrastructure should not be underestimated especially by enterprise-sized companies, where business processes and component interdependencies have reached an unprecedented level of complexity. We believe that autonomy and self-awareness of building blocks will be a key to the future design of information systems and cloud platforms.

ACKNOWLEDGEMENTS

This research has been supported by the UBIWARE project, funded by TEKES, and the industrial consortium of Metso Automation, Fingrid and Inno-W. The preparation of this paper was partially funded by the COMAS graduate school.

REFERENCES

Bellifemine, F. L., Caire, G., and Greenwood, D., 2007. *Developing Multi-Agent Systems with JADE*. Wiley.

Cercone, N.; Lijun Hou; Keselj, V.; Aijun An; Naruedomkul, K.; Xiaohua Hu, 2002. "From computational intelligence to Web intelligence," Computer , vol.35, no.11, pp. 72-76, Nov 2002.

Foster, I.; Yong Zhao; Raicu, I.; Lu, S., 2008 "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop GCE '08, vol., no., pp.1-10, 12-16 Nov. 2008.

Guazzelli, A., Zeller, M., Lin, W. and Williams, G., 2009. PMML: An Open Standard for Sharing Models. The R Journal, Volume 1/1, May 2009.

Hayes, B. 2008. Cloud computing. *Commun. ACM* 51, 7 (Jul. 2008), 9-11. DOI= http://doi.acm.org/10.1145/1364782.1364786

Kaykova, O., Khriyenko, O., Kovtun, D., Naumenko, A., Terziyan, V., and Zharko, A., 2005. General Adaption Framework: Enabling Interoperability for Industrial Web Resources, In: International Journal on Semantic Web and Information Systems, Idea Group, Vol. 1, No. 3, pp.31-63.

Kephart, J. O. and Chess, D. M., 2003. The vision of autonomic computing., *IEEE Computer*, 36(1):41–50.

Nikitin S., Katasonov A., Terziyan V., 2009. Ontonuts: Reusable Semantic Components for Multi-Agent Systems, In: Proceedings of the Fifth International Conference on Autonomic and Autonomous Systems (ICAS 2009), April 21-25, 2009, Valencia, Spain, IEEE CS Press, pp 200-207.

PMML, 2009. Data Mining Group. PMML version 4.0. WWW, URL http://www.dmg.org/pmml-v4-0.html

Rimal, B-P; Choi, E; Lumb, I, 2009, "A Taxonomy and Survey of Cloud Computing Systems," INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on , pp.44-51, 25-27 Aug. 2009.

Terziyan, V., 2003. Semantic Web Services for Smart Devices in a “Global Understanding Environment”, In: R. Meersman and Z. Tari (eds.), On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, Lecture Notes in Computer Science, Vol. 2889, Springer-Verlag, pp.279-291.

Terziyan, V., 2005. Semantic Web Services for Smart Devices Based on Mobile Agents, In: International Journal of Intelligent Information Technologies, Vol. 1, No. 2, Idea Group, pp. 43-55.