Towards a Framework for Policy-Oriented Enterprise Management

Matthias Kaiser
SAP Research Center Palo Alto
SAP Labs LLC, 3410 Hillview Ave
Palo Alto, CA 94304, U.S.A.

Jens Lemcke
SAP Research, CEC Karlsruhe
SAP AG, Vincenz-Prießnitz-Str. 1
76131 Karlsruhe, Germany

Abstract

Service-oriented architectures have brought significant progress for more flexible realization of business processes integrating functionality from heterogeneous sources. While more and more businesses adopt the new technology it becomes obvious that many questions are still not addressed to make it keep its promises, especially in the area of human efforts involved in business process composition. We introduce a framework for a possible next generation enterprise software based on, but going beyond that of service-oriented architectures utilizing logic programming taking advantage of formalized explicit policies as substantial constituents of enterprise systems.

Introduction

The use of software to help manage businesses and enterprises has a more than 30 year tradition, going through a number of changes due to hardware advancements and new software paradigms (Buck-Emden 1999). However, systems designed in the 90s were too inflexible to enable efficient adaptation of functionality to customer company demands. The result has been the development of service-oriented architecture (Woods 2003). While this approach is more flexible, it lacks properties which are critical for use by businesses. We propose a further development we call *policy-oriented enterprise management* (POEM). At the heart of this approach are, as the name suggests, policies, perceived as regulations, constraints, and contracts "steering" the behavior of, e. g., a company (Kaiser 2007).

In this short paper, we outline the major ideas behind this approach, its benefits as compared to alternatives—prominently service-oriented architectures—and describe some work currently underway to realize this initiative.

The Idea of Policy-Oriented Enterprise Management

In less than a nutshell, policy-oriented enterprise management is computational logic applied to enterprise management software. If an enterprise or an organization of any kind is to be managed, the following knowledge is necessary for a satisfactory handling of this task:

- knowledge of objects subject to management, their properties and relations;
- knowledge of policies which regulate and constraint relations and properties of objects;
- situations which are constellations of objects due to their properties and relations at a certain point in time;
- goals in form of situations which are chosen to be realized through change of initial situations; and
- certain facilities to change the properties and relations of objects taking into account policies and thus realizing new situations or goals.

If we can formalize the knowledge outlined above in a declarative way then enterprise software controls the behavior of the system utilizing a generic reasoner throughout all processes that are comprised by the operations to virtually "run" the business.

This approach is not strictly new, but has been formulated in the paradigm of logic programming (McCarthy & Hayes 1969). However, the application of this paradigm for the management of complex systems such as for the management of enterprises—or at least substantial parts of it—has not been practically implemented yet.

There are a number of benefits that justify such an idea, some of the more prominent are:

- software is realized from specifications produced by business experts rather than actual programming by software developers;
- a declarative approach facilitates uniform representation across domains and functionality empowering the adherence to the standards of formalizations;
- the explicit declaration of knowledge greatly improves understandability and maintenance, promoting dynamic adaptation for functional changes or simulations of innovative ideas in enterprise management; and
- results of decisions or business process realizations can be verified and proven on the basis of declarations and the way they were used by the control mechanisms of a reasoner.

Our vision is encouraged by multiple trends happening at a rapid pace, such as the development and implementation of service-oriented architectures (Bieberstein *et al.* 2005; Petrie & Bussler 2003), the advancements in semantic Web technologies, increased efforts in standardization of for-

mal tools to express and transform knowledge, the formation of ecosystems to distribute software and integrate open source as well as third party commercial solutions into enterprise management systems, and the increasing costs and complexity to manage software developed on the traditional programming paradigm (Grosof 2007; Margolis 2007; McComb 2003). However, we are well aware that long-term research seems still necessary to make this "dream" a reality.

Business Process Emergence in POEM: An Illustrative Example

Let us try to exemplify our vision with the following enterprise story.

Initial Situation

We take the perspective of John's Tire Center, a car tire dealer, who takes an order of four high-performance GS21 tires from Dave, a private customer. Unfortunately, John's Tire Center does not have the requested four GS21 tires on stock.

Goal

A business expert from John's Tire Center wants the system to achieve the business goal of billing Dave for his order of four GS21 tires.

Expected Business Process

Because it does not have the requested tires in stock, John's Tire Center contacts the tire manufacturer, Best Tire and Rubber, with a purchase order. Best Tire and Rubber offers special conditions for large orders, so John's Tire Center orders 100 GS21 tires, instead of the four needed for Dave. Best Tire and Rubber delivers the 100 tires and sends an invoice to John's Tire Center. John's Tire Center pays the bill and ships the four requested GS21 tires with a customer invoice to Dave.

General Policies

We envision that a policy-oriented enterprise management system would be shipped with a set of common policies such as contemporary business software systems are shipped with some pre-tailored best practice business process templates. Those general policies would, e. g., state:

- A bill can be issued for a shipment.
- A shipment can be executed from stock.
- If a shipment cannot be fulfilled from stock, the goods to be shipped must be acquired from an appropriate supplier.
- Goods will be procured from a supplier only if needed.
- A bill from a supplier has to be paid.
- No shipment is done without a prior order.

Specific Policies of John's Tire Center

Like a business software customer has to configure the purchased system by adding their master data and adapting the best practices to their business needs today, in our example, there exist policies specific to the enterprise of John's Tire Center:

- We procure any types of tires only from Best Tire and Rubber.
- We procure GS21 from Best Tire only in amounts equal to or larger than 100 pieces in a single purchase order (due to special deals).

Transforming Goals to the POEM System

In order to make the goal meaningful to the policy-oriented enterprise management system, it needs to be identified in a way that it can be decomposed into sub goals to be matched by services regulated by policies. This results in a *business* specification of the goal which can be transformed into a *technical* use case description. After refinement, an actual use case is derived in an appropriate form for processing, why may range from BPMN to first order predicate calculus (FOPC). The whole transformation process is supported by the system through proposing matching services and policies. A planner helps to organize the sequence of sub goals, and assigns services to realize them under given policies or constraints.

How POEM Works with Goals and Policies

Policies and situation descriptions are treated as logically *satisfied* statements. For example, the fact that there is an order of four tires from Dave is known. Also, the constraint is known that each ordered item not on stock must be procured. A goal is represented as a logically *unsatisfied* statement. For example, it is not yet true that there is a bill for Dave's order. Technically, the goal constitutes an inconsistency in the logical data base. Executing the plan generated for this goal resolves the inconsistency. Thus, instead of composing business processes prescriptively—as in today's service-oriented architecture approaches—they emerge in policy-oriented enterprise management based on declared goals, available functionality (services) and domain-relevant policies, all declared, which can be changed dynamically as required.

The Proper Treatment of Policies

While research and development has focused mainly on the description and utilization of services as the "active" pieces of software that change objects, policies as directives, and constraints of changes have in our opinion not received due integration yet. They play usually a minor role in service-oriented architectures, although more research is on the way (Tonti *et al.* 2003; Uszok *et al.* 2004). As has been shown in past endeavors in knowledge representation, the acquisition, representation and maintenance of knowledge—such as policies from real world scenarios—poses special challenges to humans and machines. In the following, we will outline some directions towards the treatment of policies for enterprise management.

Acquisition and Representation of Policies

The first question we are confronted with is the acquisition of policies in a way that we can explicitly represent them for reasoning in a software environment. In order to be able to

do this, it is often necessary to "translate" policies in complicated wording into a formal representation in some kind of declarations. Ideally, we would like to arrive at a representation which can easily be obtained from the original form, and which can be represented to be processable by the software and easy to understand by a human at the same time.

One interesting venue which we think could bridge the gap between ordinary expression of policies and a formal representation accessible and usable for automated reasoning is the reformulation of policies in controlled natural language (Fuchs, Kaljurand, & Schneider 2006; Schwitter 2002). A controlled natural language is a restricted subset of a natural language—such as English—which is defined by a controlled grammar and lexicon to ensure unambiguous and precise declarations. This way, a controlled natural language is really a formal language that can be processed by the computer and used to reason over the content expressed. Such a representation can easily be transformed into a commonly used logic form such as first order predicate calculus as input for a reasoning system. On the other hand, it is close enough to a natural language, and thus a user can read it without any training, and write it with comparatively little effort. Intelligent authoring tools may support the user to avoid grammatical, lexical or orthographic mistakes through instant detection, and the possibility for easy correction, including assistance for the incremental extension of the language itself. A controlled natural language could not only be used to mediate information between humans, machines, and the other way round—by means of automated generation of controlled natural language—but we can imagine to use controlled natural language even as the "lingua franca" to communicate between software and hardware components of a system or infrastructure. The big advantage of such an approach lies in the understandability of information exchanged between components of a system such as in an enterprise infrastructure.

Maintenance of Policies

Maintenance has to ensure the adequacy and validity of policies at any given point in time. This may be particularly challenging in a dynamic environment. We distinguish the following prominent cases for policy maintenance. First, policies are out of date. Second, policies are inconsistent (conflicting). And third, policies do not cover a relevant circumstance in the environment.

But also, efficiency how policies can be used is important for real world applications. This includes considering the computational complexity regarding reasoning. For this, modularity and scope of policies could be evaluated and utilized to optimize their computation. In addition, a caching approach of explicit factual policies may bring advantages compared to implicitly inferring policies all over again.

In the next section, we outline a framework for efficient policy-oriented enterprise management motivated by the above questions.

Efficient Handling of Policies in POEM

Usually, enterprises are highly structured entities, organized in departments where individuals perform tasks according to their role assignments, etc. Consequently, many policies are relevant for certain substructures of the enterprise while others are of more general nature. This fact is the starting point for our proposed policy organization which follows the structuring of a corresponding enterprise model. So, policies have scopes determined by the roles which they impact. We reflect this with a model of an enterprise as an agent population. Every agent is assigned one or more roles but may have "virtual" personal features if required. So, we can specify for each agent which policies it must adhere to.

We can think of agents as representatives of real enterprise components, prominently people, fulfilling their tasks—or at least some of them—while providing all they do transparently to human inspectors who might approve or modify behavior as they see fit. Agents are responsible to analyze situations represented as states of objects, detect relevant circumstances according to their domain of responsibilities, specify goals in response to those circumstances, which consequently will be achieved through the generation and invocation of appropriate business processes, orchestrated from services as constituents of a service-oriented architecture. Every agent consists of the following principle components:

a forward chaining rule system consisting of rules where their antecedent matches with certain circumstances in a situation while their consequences describe a high-level goal which is to be achieved in case the circumstance arises;

a planner which, if supplied with a high-level goal such as provided by the consequence of a rule for situation-relevant circumstance detection, will trigger a plan generation to produce as result a state in which the high-level goal is satisfied;

a set of facts and rules in form of a logical program which empower the planning strategy;

a set of constraints in form of policies that regulate which subgoals on the way to achieving the high-level goal and including the high-level goal itself must be obeyed.

This approach to model the agents as active "citizens" of a "virtual enterprise" is very similar to those described in (Kowalski 2005). Every agent needs to take into account a subset of policies which makes scalability of reasoning in presence of those policies much more feasible. Agents can be personalized, distributed, and easily adapted, and are thus in harmony with the service-oriented architecture approach.

In order to function correctly, regulating policies must be kept in a state of sufficiently precise reflection of the real world. Of special importance is the elimination of inconsistencies, at least the detection thereof for further human attendance. It is crucial to note, though, that policies in different subdomains may contradict. For example, one department may have different standards than another. To ease the coping of these phenomena is another good reason to divide the policy set of an enterprise into subdomains, e. g., defined by roles and utilized by agents as described above.

Besides the maintenance of contradictory policies, "holes" of situational circumstances which are not dealt with effectively—due to missing or defective policies—is of rel-

evance. Incremental policy acquisition on the basis of accurate and understandable justifications and explanations of agent actions seems a good starting point for filling those "holes," though research is necessary to realize a practical methodology.

Besides qualitative maintenance of policies, quantitative issues may also be highly critical. Crucial is the scalability of reasoning on the basis of policies. For example, it might be possible to reason correctly on a policy set over a number of reasoning steps. On the other hand, one could just explicitly write a policy which states effectively the result of the reasoning process. This may be relevant if this reasoning procedure occurs frequently and threatens to become a scalability hazard.

Of Agents and Humans

If a system takes over responsible tasks which may have a profound impact on the real world, the liability of the processes performed lies, of course, with the people who "set the system to work." Therefore, it is of paramount importance to offer accurate and efficient human-machine interfaces. We have developed an approach to equip the outlined agents with intelligent user interfaces we call "proactive user interfaces" (Kaiser & Mueller 2005) to facilitate the communication and particularly the cognitive support for the user to assist in the supervision of a policy-oriented enterprise management system. A proactive user interface consists of four main components:

Situation analyzer. This component reports which situational circumstances are relevant for an agent for further actions.

Goal recommender. Reports goals chosen and recommended to achieve as reaction on the detected situational circumstances.

Guide. Communicates a synthesized plan to achieved a recommended and accepted goal.

Explainer. Justifies the chosen process steps, which policies are relevant, and reports.

In addition, more user interfaces are needed, e. g., for maintaining data and policies, which will be subject of further research.

Conclusions

In this short paper, we have outlined our fundamental views on further development of current enterprise management systems which are formulated and pursued in our project policy-oriented enterprise management (POEM). We focus on the realization of our objectives utilizing computational logic applied to policy acquisition, representation, and utilization on top of a service-oriented architecture.

References

Bieberstein, N.; Bose, S.; Walker, L.; and Lynch, A. 2005. Impact of service-oriented architecture on enterprise systems, organizational structures, and individuals. *IBM Syst. J.* 44(4):691–708.

Buck-Emden, R. 1999. *The SAP(R) R/3 System: An Introduction to ERP and Business Software Technology*. Addison-Wesley Professional.

Fuchs, N. E.; Kaljurand, K.; and Schneider, G. 2006. Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces. In *Proceedings of 19th International Florida Artificial Intelligence Research Society Conference, Melbourne Beach, Florida, USA (11th–13th May 2006)*, 664–669. The Florida Artificial Intelligence Research Society.

Grosof, B. 2007. Commercializing semantic web: Rules, services, and roadmapping. Invited Keynote Presentation (1-hour) at the 1st European Semantic Technology Conference (ESTC-2007), Vienna, Austria.

Kaiser, M., and Mueller, C. 2005. The shopping scout: A framework for an intelligent shopping assistant. In *Proceedings of the 2nd International Conference on e-Business and Telecommunication Networks, Reading, UK*.

Kaiser, M. 2007. Toward the realization of policy-oriented enterprise management. *IEEE Computer Society Special Issue on Service-Oriented Architecture* forthcoming.

Kowalski, R. A. 2005. The logical way to be artificially intelligent. In Toni, F., and Torroni, P., eds., *CLIMA VI*, volume 3900 of *Lecture Notes in Computer Science*, 1–22. Springer.

Margolis, B. 2007. *SOA for the Business Developer: Concepts, BPEL, and SCA*. Mc Press.

McCarthy, J., and Hayes, P. J. 1969. Some philosophical problems from the standpoint of AI. In Meltzer, B., and Michie, D., eds., *Machine Intelligence*, volume 4, 463–502. American Elsevier.

McComb, D. 2003. *Semantics in Business Systems: The Savvy Manager's Guide*. Morgan Kaufmann.

Petrie, C., and Bussler, C. 2003. Service agents and virtual enterprises: A survey. *IEEE Internet Computing* 7(4):68–78.

Schwitter, R. 2002. English as a formal specification language. In *DEXA Workshops*, 228–232. IEEE Computer Society.

Tonti, G.; Bradshaw, J. M.; Jeffers, R.; Montanari, R.; Suri, N.; and Uszok, A. 2003. Semantic web languages for policy representation and reasoning: A comparison of KAoS, rei, and ponder. In Fensel, D.; Sycara, K. P.; and Mylopoulos, J., eds., *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, 419–437. Springer.

Uszok, A.; Bradshaw, J. M.; Jeffers, R.; Tate, A.; and Dalton, J. 2004. Applying KAoS services to ensure policy compliance for semantic web services workflow composition and enactment. In McIlraith, S. A.; Plexousakis, D.; and van Harmelen, F., eds., *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, 425–440. Springer.

Woods, D. 2003. *Enterprise Services Architecture (O'Reilly Field Guide to Enterprise Software)*. O'Reilly Media, Inc.