From composition to emergence

Towards the realization of policy-oriented enterprise management

Matthias Kaiser
SAP Research Palo Alto, U.S.A.

August 20, 2007

Today, service-oriented architectures as basis for the composition of business processes are widely seen as the state-of-the-art approach to realize flexible, extendable enterprise management. However, a number of problems how to efficiently use this architecture to compose applications to support business goals is still a hard problem requiring specific expertise as well as tedious human involvement.

In this article, we motivate and outline a new approach towards goal-driven business process composition based on the "enterprise physics metaphor". On the foundation of formally stated business goals, descriptions of Web services and the formalization of business policies, we explain how business processes capable to achieve stated business goals can be generated utilizing generic, logic-based strategies.

Introduction

Traditional enterprise software systems

Enterprise software systems are used today for the management of diverse operations in big and medium businesses. All data and processes occurring in such a complex operations landscape are modeled in modules which are part of a unified "software reflection". The probably best known enterprise software system is SAP's R/3 system built on the foundations of a three-tier architecture (Buck-Emden, 1999). This architecture includes the following components as depicted in figure 1. The *computing infrastructure* consists of database servers managing documents, meta data and master data. The *integration and application infrastructure* consists of application servers that host application software. *Business applications* access the application servers as clients for user-system interaction.

The functionality of enterprise software systems is divided into large modules, each of these modules containing typically support for a whole business department. Modules making up the system are, for example:

- Business Intelligence Suite (Information Warehouse and Analytics)
- Customer Relationship Management (CRM)
- Supply Chain Management (SCM)
- Supplier Relationship Management (SRM)
- Human Resource Management Systems (HRMS)
- Enterprise Resource Planning (ERP)
- Product Lifecycle Management (PLM)
- Exchange Infrastructure (XI)
- Enterprise Portal (EP)
- Knowledge Warehouse (KW)

The named modules are available in the current R/3 system of SAP. To give an impression on the complexity of these modules, figure 2 shows the main business processes provided in the enterprise resource planning module of SAP R/3 (Hartmann, 2005).

Problems of traditional enterprise software systems

While this approach has been adopted widely and helps run businesses successfully, it has also a number of serious problems where the following are most promi-

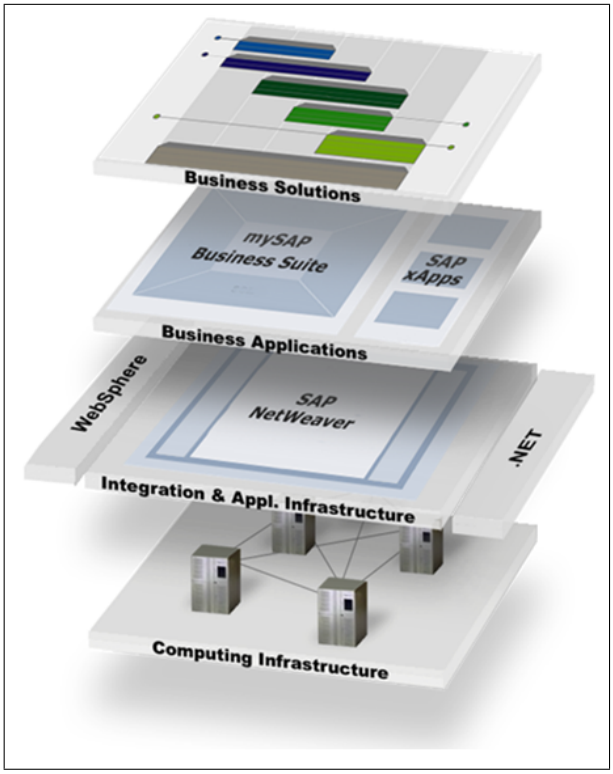

Figure 1: SAP's three-layered architecture. The top layer labeled "business solutions" is not counted as one of the three architecture layers. It depicts the flexible applicability of SAP solutions in various business domains.

nent (Grant et al., 2006).

First, systems tend to be very monolithic. The envisaged advantage of integrated systems becomes a burden, because relationships between modules become too complex to manage. This is generally referred to as the problem of *system decomposition*.

Second, systems are hard to implement because of complex interdependencies and requirements for successful interoperability. Furthermore, systems are used in many different types of organizations. Thus, they are heavily parameterized. To adjust all parameters appropriately to organizational needs is a very difficult and time-consuming implementation process that often over lasts for many periods of refinement cycles. This is often called the *feature interaction* problem.

Third, the structure of systems essentially reflects the business-functional structure of an organization. That is, a module reflects the business function of a whole department. This makes it hard to realize processes crossing traditional business-function boundaries. It makes the reaction to changing business processes or ad-hoc business process demands almost impossible. In fact, the constraints of modules on the operation of businesses may have the effect that the departments try to align their operations with the capabilities of the software modules they use!

From monolithic systems to service-oriented environments

During using such complex monolithic enterprise software systems, a number of requirements emerged as guidelines to overcome existing problems. Among these the following are prominent (see Bieberstein et al., 2005; Margolis, 2007).

- Realize system components which are flexibly composed into applications.
- Aid composition and parameterization of components for composition.
- Provide possibility for easy adaptation to specific user needs regarding business structure to make changes of software to new demands manageable.
- Allow for the possibility to integrate third-party or customer- proprietary solutions into the system.

Service-oriented architecture

The answer to these demands is called *service-oriented architecture* (SOA). This new architecture is founded on the objectives to support rapid design, development, discovery and consumption of standardized services (Erl, 2005). In particular, if we use services in the context of enterprise software, we term them enterprise services to emphasize specific properties they have to exhibit.

Enterprise services are composite Web services that support a process step meaningful in a business process. They are composed from lower level services and can be seen as wrappers of business objects, providing business logic for their manipulation. Enterprise services are de-

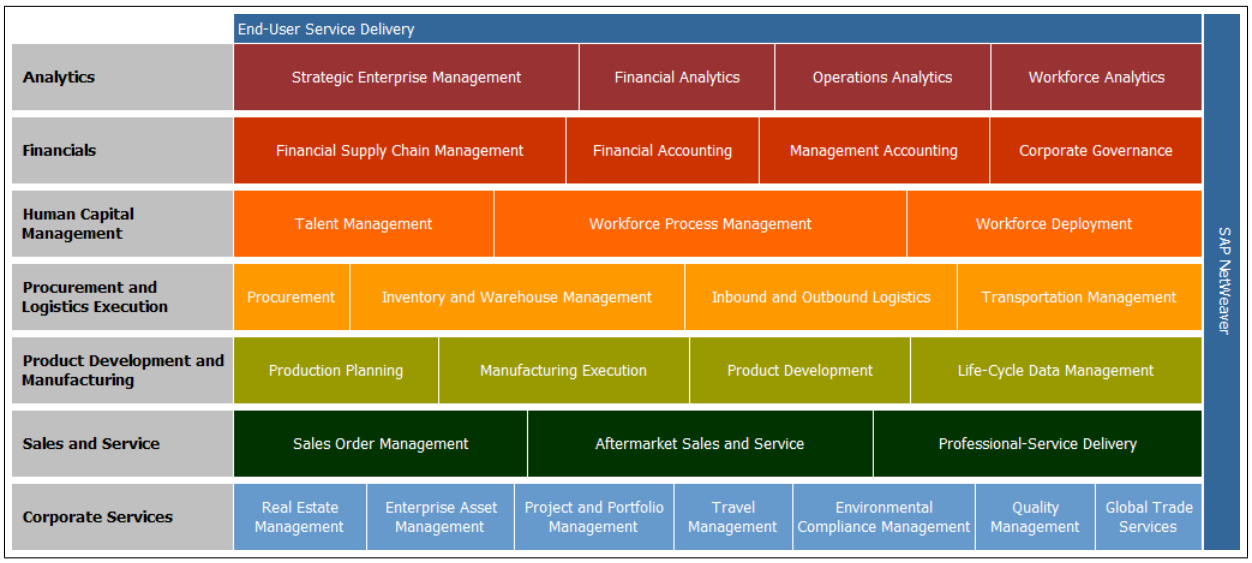

Figure 2: mySAP ERP Solution Map. The boxes to the left name business process categories. Each box in the middle section represents a *main business process* of the enterprise resource planning application module of SAP's R/3 system. A main business process is a grouping of concrete *business processes* serving a similar purpose. Each business process can have different *configuration variants* in the different areas of their application. More detail can be found on line at `http://www.sap.com/solutions/businessmaps/`.

scribed according to their requirements, deliveries and behavior in a standardized way (Woods, 2003). A *business object* is an identifiable business entity (like customer, or product). It can have complex structure and is manipulated by services through interfaces. A *business process* is a sequence of business-relevant process steps towards a business goal. The *enterprise service repository* provides information about services and business processes including business object specifications, service descriptions, and business processes.

The flexibility of this kind of architecture is obvious if we see how business processes are composed (see figure 3):

1. Given a business process scenario,
2. Design a business process model, and
3. Derive a choreography for the available services to instantiate the business process model.

Enterprise services are used to compose applications. An application is built by combining multiple enterprise services. A composite application consists of functionality drawn from several different sources within a service-oriented architecture. The components may be individual Web services, selected functions from within other applications, or entire systems whose outputs have been packaged as Web services (often legacy systems). SAP's take on service-oriented architecture is called enterprise SOA. It is depicted in figure 4.

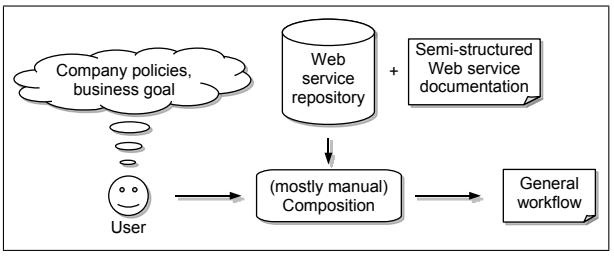

Figure 3: Business process design in service-oriented architecture. Business experts carry out the manual business process composition procedure. The essential knowledge of company policies, business goals and instructions on Web service usage resides in a semi-structured form in the heads of the business experts and text documents.

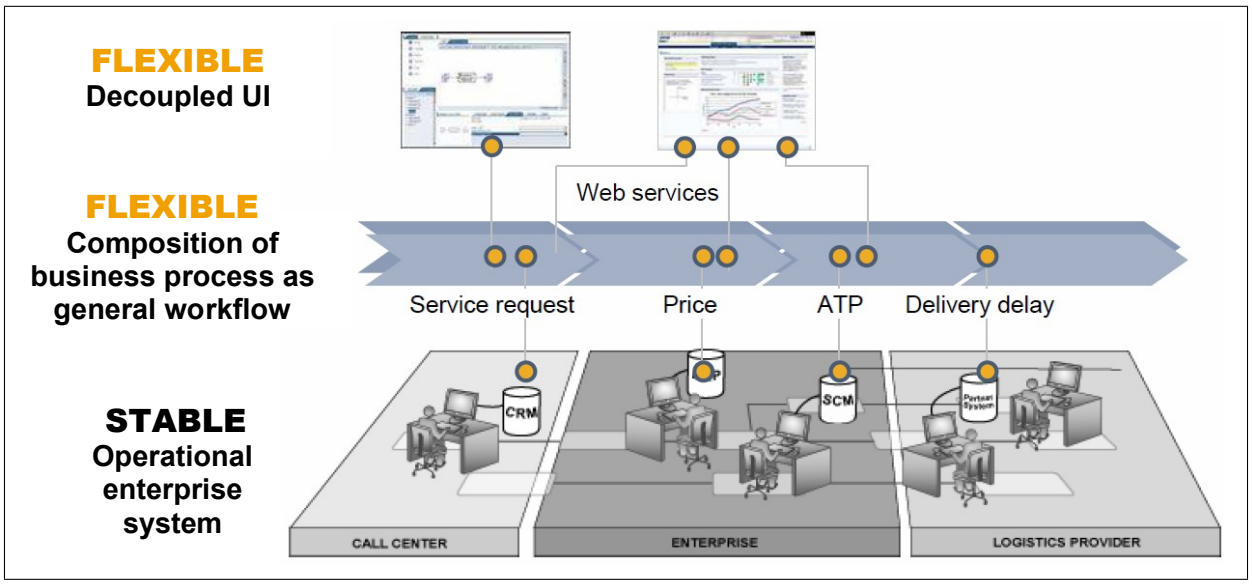

Figure 4: SAP enterprise SOA. The service-oriented approach allows to uniformly expose monolithic, legacy systems as well as modern, service-oriented systems as service interfaces to the system user (lower layer). Composite services can be built based on this uniform Web service representation. Composite services can be arranged as business processes that provide a unified view on the systems and processes in a company (middle layer). People-centric user interfaces can be built flexibly on the unified business process view leveraging the different legacy and service-oriented systems (upper layer).

Discussion of service-oriented architecture

The approach outlined above has without doubt significant advantages over the traditional enterprise software systems architecture. For example, existing applications can easily be modified and extended through the injection or extension with new services. In addition, new and proprietary business processes can be composed from reusable, available services or based on newly created services. This reduces the time for customizing and configuring of new and auxiliary applications.

While the service-oriented architecture for business software development has undoubtedly quite a number of advantages, there happen to be some serious issues as well:

- Processes have to be composed in advance to meet potentially relevant business goals. This makes it hard to react quickly and flexibly to meet a business goal not foreseen with an ad-hoc business process.
- The maintenance effort of composed processes is very high. This is especially true when it comes to ensuring the consistency with a changing business process environment, e. g., if business processes which are constituents of other processes change.
- If services themselves are modified or more efficient services to realize a business process step become available, re-tailoring of business processes is tedious.

There are two ways to react to these issues. One is to further refine the service-oriented approach as Papazoglou et al. (2007) do. The other is to develop a new paradigm to advance the service-oriented approach (Petrie and Bussler, 2003; see also Steffen and Narayan, 2007; in this issue). This is what we describe in the following section.

Policy-oriented approach

In every business, there is usually quite a number of regulations, guidelines, policies, rules or whatever we want to call them that express constraints on the working of that

business, how business processes may work, what is regarded as beneficial outcomes and results, etc.

If we can take for granted that semantics of services are described using standardized means and that rules and policies which are ensuring the correct operations of the system can be declared in an unambiguous manner, we could actually formulate a business goal in a declarative way which can be satisfied by composing business processes in a deductive way, constrained by policies. Instead of trying to compose business processes from single components, we can basically enter a business goal from which an appropriate process to satisfy this goal is produced (Margaria and Steffen, 2007; McIlraith and Son, 2002; Steffen et al., 1997). This is obviously a more natural way to generate business processes, at least from the perspective of a business user.

Enterprise physics metaphor

This approach has been called the *enterprise physics metaphor*. The enterprise physics metaphor bases on the following hypothesis. Business objects are analogous to physical objects; policies are analogous to physical laws; and services change objects within the constraints of policies. Figure 5 depicts the metaphor using the billiard game as an example for a physical system mapped to the enterprise world.

In the enterprise physics metaphor, a business process is realized as follows. A business goal is described by properties and relations of and between business objects. This corresponds to a certain arrangement of balls after one or more strokes have been performed. The current business situation is known, i. e., the current properties and relations of and between business objects. In the metaphor this means that the positions and properties of the balls are known.

Business policies are declared. They correspond to natural laws like gravity and kinetic laws in the metaphor. Processes are composed by orchestration of services to transform the current situation into a situation in which the declared goal is true under the declared policies. In the metaphor, services correspond to the kinds of strokes which are performed to change the constellation. The series of strokes corresponds in a planful manner to the business process which has been planned on the basis of the goal and the current or initial situation.

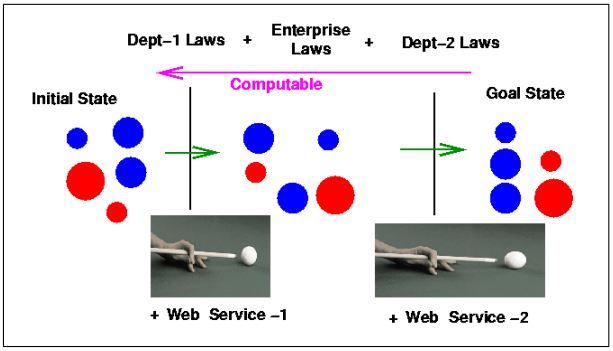

Figure 5: Enterprise physics metaphor. Each billiard ball is a physical object. The positioning of balls at a specific time is analogous to a specific state of the world. Each stroke of a billiard cue from a specific angle with a specific strength is analogous to a service capable of changing the state of the world. The balls move as a result of a stroke respecting the physical laws of momentum and conservation of energy. (Source: `http://logic.stanford.edu/talks/deriwest/sap-executive/`)

This procedure is depicted in figure 6.

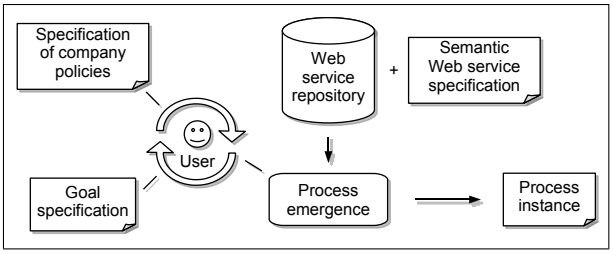

Figure 6: Business process design in policy-oriented architecture. In contrast to the service-oriented approach in figure 3, company policies, business goals and Web service specifications are given in a machine-readable way. Subsequently, the user can be strongly supported in the design of a desired business process instance.

Discussion of the enterprise physics approach

Comparing this approach to the shortcomings of the service-oriented architecture in place today, we can see the following benefits.

- Processes can be “developed” fast without human

programming. Human programmers do not have to deal with the complexities of process development, customization and integration.

- Processes that need to meet flexibly changing ad-hoc business goals can be realized quickly.
- The whole procedure of business process design completely adheres to a consistent interpretation of the business policies. No interpretations are conjectured by programmers.
- Processes can be proven with regards to their validity on the basis of theorem proving. For example, Braun et al. (1998) propose to apply model checking for this purpose.
- Knowledge about company policies is well maintained in the system, instead of being scattered in the heads of experts.

Of course, a solution like the one outlined is impossible to implement in a general way anytime soon because of not yet fully realized preconditions. Semantic data integration and interoperability has not yet been achieved in a way which enables large scale utilization (Genesereth et al., 1997). The same holds for the consistent formalization of descriptions of services, policies, processes, human information exchange, and so on (Genesereth et al., 1997). Fundamental questions like the power of expressiveness, granularity of descriptions, scope and compliance with existing standards have to be considered in addition (McComb, 2003).

Example

In the following, we give a high-level description of an example illustrating our novel policy-oriented approach.

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

Goal. A business expert from John's Tire Center intends to have the system achieve the following business goal.

Bill Dave for his order of four GS21 tires.

Expected business process. Since John's Tire Center does not have the requested tires on stock, John's Tire Center contacts the tire manufacturer "Best Tire and Rubber" with a purchase order. Due to special conditions with Best Tire and Rubber for large orders, John's Tire Center decides to procure 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 GS21 tires initially requested together with a customer invoice to Dave.

Transformation to the enterprise physics domain. The first task is to identify this business goal in a way that it can be decomposed into its components, which will be matched with available services and policies on the basis of their description. The result will be a business specification of the goal as start for a transformation into a technical use case description. In this process, the user engages in an interaction with the system, where the system informs the user about discovered matching descriptions of services and policies as well as informing about mismatches and conflicts. So the user can refine the business goal into a goal which can be realized given the system's "enterprise physics" environment.

Once the goal has been identified, a use case description can be derived that is realized in an appropriate form and language that may range from the *business process modeling notation* (BPMN)[1] to *first order calculus*. Thus, the business goal specification is transformed into a technical specification being the foundation for the subsequent business process generation.

Behavior of the enterprise physics system. Once the example above is expressed in terms of the enterprise physics domain, an enterprise physics system reacts in the following way. The statements contained in the situation description of our example as well as general business rules end up as policies in the enterprise physics system that are *satisfied* in the logical sense. An example is the *fact* that there is an order of four GS21 tires from Dave known to the system. Another example is the *constraint* that each item ordered that cannot be directly supplied from stock must be procured from an appropriate supplier. These statements are always true for all orders that are in the system yet.

In contrast to the situation description, the goal shows up as a policy that is currently *unsatisfied* in the system. In our example, it is not yet true that there is a bill for Dave on four GS21 tires. Now, the unsatisfied policy

[1] `http://www.bpmn.org/`

constitutes an inconsistency in the knowledge base of our enterprise physics system. Such a state is not desirable. Therefore, the enterprise physics system seeks a way to overcome the inconsistency and behaves appropriately. In the case of our example, the system proposes a plan of actions that will be able to transform the current situation to a situation where the goal policy is satisfied. Thereby, the actions can be either manual tasks to be performed by human actors or service invocations of John's Tire Center's service-enabled enterprise system. Concretely, the system proposes to procure 100 GS21 tires from Best Tire and Rubber, send the four requested tires to Dave and handle the billing as laid out in the expected business process section above.

Some technical challenges on the way to the realization of the approach

In order to actually realize such an approach, a number of technical challenges, many not unlike those relevant for successful deployment of a service-oriented architecture, are to be mastered. In the following, we mention some of them.

Semantic data integration and interoperability. For an approach like ours to be successful, it is critical that we can represent and describe involved entities in a consistent form, which is expressive enough and yet performant and extendable to be usable in a real-world business environment. Many standards have been proposed for all kinds of purposes. These are usually XML-based languages.[2] We have to analyze which ones can meet our needs and have the scope required stretching over descriptions of services, policies, processes, human information exchange, and so on. In parallel to the development of standardized languages for modeling like the *Web services business process execution language* (WS-BPEL),[3] the *process specification language* (PSL)[4] or the *Web services description language* (WSDL),[5] we suggest work on a unified "lingua franca" between humans and machines considering alternative logics as temporal or modal logic, and possibly even between machines for a better inspection by humans using a controlled natural language (Fuchs et al., 2006).

Dynamic Web service integration. A distinguishing property of our approach is the emergence of a business process on the basis of a usually user-supplied business goal. This business goal is realized by a business process which emerged from the integration of single services into an appropriate sequence (or more complex arrangement). This emergence is performed by an appropriate planning functionality. So we need a planner powerful enough to build business processes from services including service discovery, consumption, composition and verification. In real-world applications (Burstein and McDermott, 2005), it seems not enough to describe services solely by their functional properties. Rather side effects and higher-level information may be crucial. We suggest work towards more real-world appropriate service specifications.

Crucial is that the planner observes policies imposing business-critical constraints on the composition, i. e., planning task (Tonti et al., 2003).

An important issue is efficiency. Not every business goal can only be realized by a brand-new business process. Rather, the reuse of already available business processes and their adaptation is an equally important problem of very high real-world value.

Enterprise policy research. A central topic is, of course, research into enterprise policies which appear in various kinds and forms. Different enterprises and even departments have different policies. Thus, dynamic business policy integration is necessary. Goals must be met while at the same time complying with individual policies. Inconsistency in data and rules, and incomplete information are challenges.

Policy acquisition, formalization, modification, and monitoring are further essential tasks. How can the user state correct policies, monitor their behavior and intervene in cases of conflicts? Can policies be derived automatically from existing processes?

Although this research has just seriously begun, there are already usable results from computational logics, operational research, even research in legal expert systems and related subjects and results can be already used profitably in real-world scenarios (Tonti et al., 2003; Uszok et al., 2004).

[2] XML: extensible markup language, `http://www.w3.org/XML/`

[3] `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`

[4] `http://www.mel.nist.gov/psl/`

[5] `http://www.w3.org/TR/wsdl`

Human-machine interaction. An especially important matter is how human users will work with a system utilizing our approach. What are the benefits, what are the challenges when humans interact (Kaiser and Mueller, 2005)? While the challenges mentioned so far are not exclusive for a policy-oriented enterprise management paradigm, the benefits and interaction scenarios are, and therefore we will dedicate somewhat more elaborate outlines for this. Prominent questions seem to be the following.

- *Goal identification.* Goals of business processes are often complex and difficult to state by a user correctly. How can the system help the user to determine correct (potentially) achievable goals on the basis of an initial situation and available resources (services, policies...)?
- *Policy selection, acquisition and monitoring.* Policies are often not explicitly formulated in the system. How can they be acquired? How can they be monitored for correctness? How can the user intervene in case of conflicts of policy application?
- *Explanation of synthesized processes.* How can the system explain or prove a generated process or specific decision for process steps to a human user?

In the next section, we would like to outline a promising way how our approach could be supported by an assistant facilitating human-machine interaction to collaboratively realize business processes, but at the same time benefiting from the policy-oriented knowledge available for a multitude of support, recommendations, optimizations, etc.

Using the POEM approach in a business environment

Policy-oriented enterprise management (POEM)[6] is the name of an ongoing research project between SAP Research[7] and the computational logic group at Stanford University.[8] The POEM project exploits the ideas of the policy-oriented approach in the business environment. We will briefly outline in which way our approach can support a user in an everyday business operation and consequently what an interface, or, as we call it, for its proactive features, interface assistant has to provide. The overview below shows a short summary of features most relevant in a user interaction with a complex continuously adapting software system for cognitive support:

- Achieving "real-world awareness" by means of situation description, especially of elevant situation constituents, i. e., indicators of problems or business opportunities
- Assistance in formulating achievable process goals based on resources and context, i. e., available services, constraining policies, role specifications of the user
- Decision support for conflict resolution, policy acquisition and monitoring,
- Explanation of generated process designs (proofs), stating which services and policies were observed and used, respectively
- Automatic generation of process design documentation.

We envisage such an assistant to consist of basically four components (see figure 7) in order to realize the features shown.

1. Situation analyzer. A situation is characterized prominently by states of plans (how far is a plan to achieve a certain user goal fulfilled) and (related to this) whether there are inconsistencies between the situation and policies which act as "laws" to be observed. The situation analyzer can report, or even alert, these special circumstances so that the user can know and act accordingly, taking advantage of the other components of the assistant outlined below. We would like to note that the situation analyzer is not only able to describe situations in the real business scenarios but can equally be used in simulated situations. Simulations of business processes to gain insight for business-process adaptations and improvement can play a decisive role to optimize business operations. and come up with new or refined policies and even help to define requirements for new services.

2. Goal recommender. Based on situation descriptions revealing the state of a business situation, specifically those situational constituents representing inconsistencies with established policies, the goal recommender can generate business goals (or subgoals thereof) which will lead to a new situation where plans are completed and/or policies are satisfied, respectively. Since goals are generated on

[6] http://logic.stanford.edu/POEM/
[7] http://www.sap.com/company/research.epx
[8] http://logic.stanford.edu/

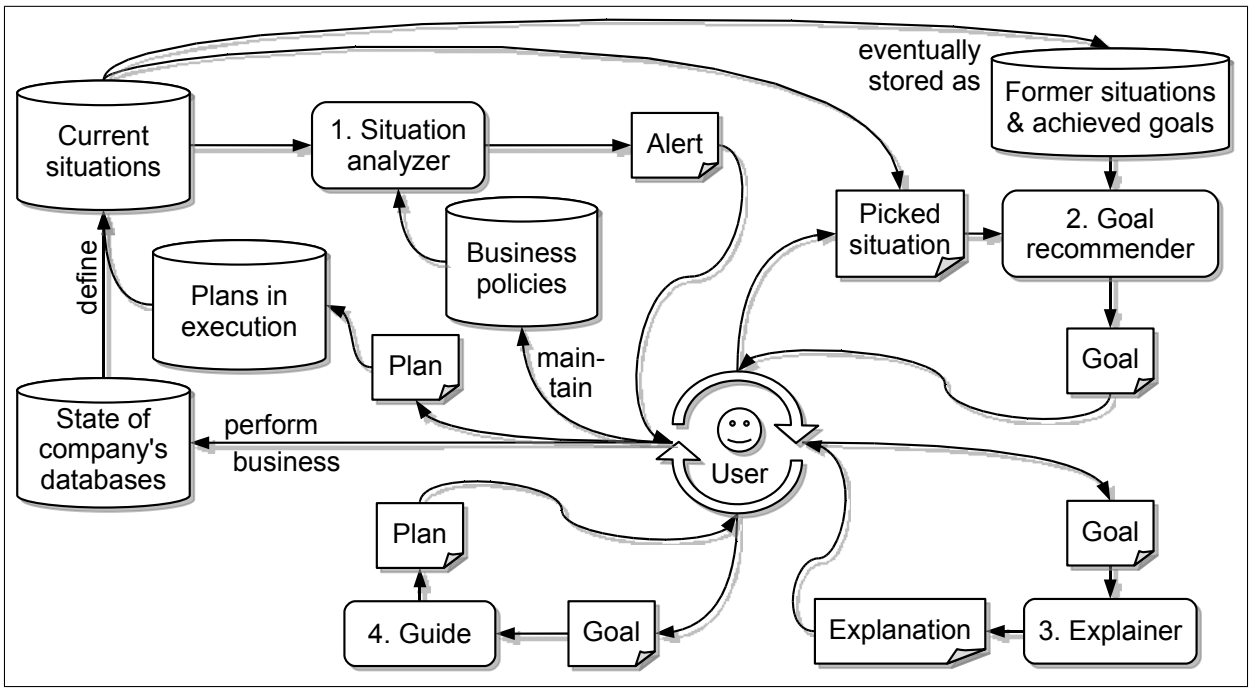

Figure 7: Interface assistant architecture. The four components in the rounded boxes are numbered corresponding to the description in the text. Document-like symbols denote inputs and outputs of the components. Database symbols represent persistent collections of data. The user interacts with the four components in an arbitrary order. Whenever the user performs its business, it changes the state of its company's database. In addition, the user manages the business policies that are evaluated by the situation analyzer. Once a plan has come to an end, its original situation plus the goal it achieved are stored in the case base of former situations forming an important constituent of the context for further interaction and is used by the goal recommender.

the basis of inconsistencies between policies and the current situation we can look at a goal the user or another "goal generating agent" brings into the system as a policy too. A goal is some state which an authority (the user) has made a law, so that the system will have to aspire (using plans) to satisfy. A goal which has been recommended will, provided it is confirmed by the user, trigger planning for its satisfaction.

3. Explainer. In a complex software system where "intelligent" operations take place it is of paramount importance that the user is able to understand the system's states and progress. This is of particular importance when the system adapts or evolves so that even experienced users can not rely on their knowledge reflecting how the system behaved previously. This can be aided in our scenario by the explainer. The explainer can justify why certain goals are recommended (their cause), what the effect of satisfying a recommended goal will be. This information must be presented in a way comprehensible by a business expert rather than in a technical manner. So The user is given the necessary information to understand the behavior (or planned behavior) of the system and she can act accordingly in an informed manner.

4. Guide. The information about *why* a goal has been recommended and *why* this goal is relevant in the current situation does not yet put the user in the position to actively support the pursuit of that goal because it is not yet explicit how to achieve it. This is where the guide comes in. The guide explicates actions which, if carried out, will help to achieve the goal. Note that each action itself achieves a subgoal which is recommended, can be explained and forms together a recursive process involv-

ing the assistants' components in the support of the user.

From plan to program: Making the idea reality

The essential objective of the POEM project is to show the applicability, value and feasibility of the use of computational logic in modern enterprise management as a next step in software development. Actually we believe that the application of computational logic can lead to a paradigmatic shift in the relation between the enterprise management and the software supporting it, which may even result in a revolutionary rather than an evolutionary step. closing the gap between business experts' understanding of their domain and the software engineers realizing appropriate software.

Today we are leveraging from the groundbreaking theoretical work in this area at the computational logic group at Stanford University to conduct studies in the applicability of computational logic as a partial or even general theoretical foundation for next generation software architectures and suites. Once we show the value of technology based on computational logic by means of prototypes working in test environments, we will proceed with considerations and necessary research to build a platform allowing feasible broad range use of this technology within a serious software system, eventually in the new generation product of tomorrow.

Acknowledgments

I would like to sincerely thank the POEM team, Michael Genesereth, Charles Petrie, Jens Lemcke and Susanne Glissmann for their helpful advice, support and encouragement!

References

Bieberstein, N., S. Bose, et al. (2005). Impact of service-oriented architecture on enterprise systems, organizational structures, and individuals. *IBM Syst. J.*, 44(4):691–708. ISSN 0018-8670.

Braun, V., T. Margaria, et al. (1998). Automatic error location for in service definition. In *ACoS '98/VISUAL '98, AIN '97: Selected papers on Services and Visualization: Towards User-Friendly Design*, pages 222–237. Springer-Verlag, London, UK. ISBN 3-540-64367-2.

Buck-Emden, R. (1999). *The SAP(R) R/3 System: An Introduction to ERP and Business Software Technology*. Addison-Wesley Professional. ISBN 201596172.

Burstein, M. H. and D. V. McDermott (2005). Ontology translation for interoperability among semantic web services. *AI Mag.*, 26(1):71–82. ISSN 0738-4602.

Erl, T. (2005). *Service-Oriented Architecture*. Prentice Hall PTR, Upper Saddle River. ISBN 0131858580.

Fuchs, N. E., K. Kaljurand, et al. (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)*, pages 664–669. The Florida Artificial Intelligence Research Society. URL `http://idefix.pms.ifi.lmu.de:8080/rewerse/index.html#REWERSE-RP-2006-026`.

Genesereth, M. R., A. M. Keller, et al. (1997). Infomaster: an information integration system. *SIGMOD Rec.*, 26(2):539–542. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/253262.253400.

Grant, David, et al. (2006). The false promise of technological determinism: the case of enterprise resource planning systems. *New Technology, Work and Employment*, 21(1):2–15. ISSN 0268-1072. doi:10.1111/j.1468-005X.2006.00159.x. URL `http://dx.doi.org/10.1111/j.1468-005X.2006.00159.x`.

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.

Hartmann, G. (2005). *Product Lifecycle Management with SAP: The Complete Guide to mySAP PLM Strategy, Technology and Best Practices*. SAP PRESS. ISBN 1592290361.

Kaiser, M. and C. Mueller (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*.

Margaria, T. and B. Steffen (2007). Ltl guided planning: Revisiting automatic tool composition in eti. In *31st Annual Software Engineering Workshop Loyola College, Baltimore, MD, USA*.

Margolis, B. (2007). *SOA for the Business Developer: Concepts, BPEL, and SCA*. Mc Press. ISBN 1583470654.

McComb, D. (2003). *Semantics in Business Systems: The Savvy Manager's Guide*. Morgan Kaufmann. ISBN 1558609172.

McIlraith, S. and T. Son (2002). Adapting golog for composition of semantic web services. URL `citeseer.ist.psu.edu/mcilraith02adapting.html`.

Mittal, K. (2006). Requirements process for soa projects, part 1 of 3: Capturing requirements for an soa application - initial requirements to build out your soa. Web page. URL `http://www.ibm.com/developerworks/library/ar-soareq/index.html`.

Papazoglou, M. P., P. Traverso, et al. (2007). Service-oriented computing: State of the art and research directions.

Petrie, C. and C. Bussler (2003). Service agents and virtual enterprises: A survey. *IEEE Internet Computing*, 7(4):68–78. ISSN 1089-7801. doi:http://dx.doi.org/10.1109/MIC.2003.1215662.

Steffen, B., T. Margaria, et al. (1997). The electronic tool integration platform: Concepts and design. *STTT*, 1(1-2):9–30.

Steffen, B. and P. Narayan (2007). From end-to-end processes to deployment.

Tonti, G., J. M. Bradshaw, et al. (2003). Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder. In D. Fensel, K. P. Sycara, and J. Mylopoulos, editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 419–437. Springer. ISBN 3-540-20362-1. URL `http://dblp.uni-trier.de/db/conf/semweb/iswc2003.html#TontiBJMSU03`.

Uszok, A., J. M. Bradshaw, et al. (2004). Applying kaos services to ensure policy compliance for semantic web services workflow composition and enactment. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 425–440. Springer. ISBN 3-540-23798-4. URL `http://dblp.uni-trier.de/db/conf/semweb/iswc2004.html#UszokBJTD04`.

Woods, D. (2003). *Enterprise Services Architecture (O'Reilly Field Guide to Enterprise Software)*. O'Reilly Media, Inc. ISBN 596005512.