Product Reasoning Services: Economic Relevance and Architectural Approaches

Dipl.-Kfm. Martin Hepp, Universität Würzburg

hepp@martinhepp.de

Contents

Citation Information

Hepp, Martin: Product Reasoning Services: Economic Relevance and Architectural Approaches. In: Dangelmaier, W. et al. (Eds.): Innovationen im E-Business. Fraunhofer-ALB, Paderborn 2003, p. 469-477. PDF file available at

www.heppnetz.de/prs-hepp2003.pdf

1. Introduction

Digital business transactions are being carried out on a vast number of diverse systems all around the world. Up to now, one single, monolithic trading and messaging platform has not evolved and will probably not exist in the foreseeable future. Instead, different industries use different systems, message formats, vocabularies (e.g. product codes), etc. It seems that we will have to deal with a heterogeneous and distributed environment for future E-business transactions.

If that is the case, those functionalities which are needed by users from various industries or using different technology should be implemented as independent services. This is for two reasons. First, they will frequently not be part of library-like standard software packages, because they might address requirements of a very specific domain. The second and more important argument for this approach is that it allows for those services to be continuously improved independently from other components' maintenance cycles. A service that calculates the fraud risk of specific business transaction might, for example, require more frequent updates with regard to the knowledge base and heuristics included than the shopping system that calls it.

This paper argues that one major obstacle on the way to fully integrated business processes is the amount of reasoning tasks which require manual interaction. It is obvious that numerous business transactions include reasoning tasks, e. g. about goods, services, or delivery details.

Reasoning is here used as a term for deriving new knowledge from existing knowledge. Examples of such reasoning tasks are the following queries:

- Can product 1 be used as a substitute for product 2?
- Does a specific product (described by unstructured text data or a vendor-specific number) belong to a given class of a classification system (eCl@ss, UN/SPSC,…)?
- Which will be the appropriate shoe size, derived from a vectorised foot shape and the desired brand?

The paper identifies economically relevant reasoning services, discusses architectural issues, and suggests suitable method calls with the respective parameters.

2. Required Services for Electronic Markets

Reasoning can be defined as a system's ability to "derive conclusions (e.g. diagnoses, designs, schedules) that are sanctioned by its knowledge" (Greiner/Darken/Santoso 2001, p. 2). The crucial difference to a mere database query is that the desired answers are not explicitly stored in the system (Greiner/Darken/Santoso 2001, p. 2), but are gained out of implicit knowledge in the knowledge base and possibly from external sources. It is even possible that one reasoning service buys advice from another reasoning service or tries to validate its own conclusions by comparison with others.

The following is a short and incomplete list of areas of application that would benefit from access to such reasoning services.

- **Agent-based E-commerce** in general does not only require an XML-framework for document exchange and semantically unambiguous encoding (cp. Glushko/Tenenbaum/Meltzer 1999), but moreover access to complimentary information or services in order to determine the best match. Agents need not only be able to buy and sell goods, but also such additional information or advice. They should thus be able to query reasoning services.

- This is especially true for **agent-based arbitrators** (Hepp 2002, p. 482-485) who try to make use out of price dispersion among different market places and thus help to increase overall market integration.

- **Reasoning about legal aspects:** An agent or a computer system might need to know whether a specific business transaction complies with the laws in the relevant region during a given period of time. This is what human actors expect when asking the legal department for advice.

- **Reasoning about risks:** The risk associated with a specific business transaction is determined by a multitude of factors, e.g. the opponents' identity, the value of the transaction, the kind of goods or services traded, the delivery address, etc. It would be useful for many applications to determine the risk of fraud or other mischief. Credit card authorisation processes are a form of such reasoning about risks, but are limited to the payment.

Reasoning services are most useful when the result to a query is determined not only by one single parameter but by a multitude of factors, to which the querying application has no adequate access. Examples are situations where the time, location, or the identity of the parties involved influence the result.

Such reasoning engines will be important components of future eBusiness architectures. This is because for a huge set of transactions, computer systems require answers to such queries, in order to be able to perform the respective transaction autonomously. The following examples focus on reasoning about products and services, though the main idea can be as well applied to further domains.

2.1. Product Comparison

Price-comparison agents, agent-based arbitrators, CRM solutions or catalogue management software need to be capable to judge whether two different products can be treated as instances of the same kind, i.e. whether the one product can be used as a substitute for a second one or whether it makes sense to list both in the same catalogue section.

This could be implemented as a method call of the following kind:

public float compareProducts (String Product1, String Product2)

The input parameters "Product1" and "Product2" could be either unstructured text descriptions or small XML-documents, depending on the application domain and the intelligence of the reasoning engine. The resulting float value will return the measure of similarity.

The appropriate threshold will depend on the application domain. A catalogue management solution will accept a lower degree of similarity, whereas an agent-based arbitrator will require a remarkably higher value.

It should be noted that it is not necessary to have one single service for all domains. Moreover, services for specific subdomains will probably evolve, e.g. "compareDigitalCameras(…)" or "compareBuildingMaterials(…)".

2.2. Product Classification / Formal Product Description

This task is performed by most catalogue data rationalisation tools and services, which need to transform existing, poorly structured and semantically ambiguous data into well-structured XML documents (e.g. BMEcat or an xCML catalogue instance). According to DOLMETSCH, US content providers claim to be able to rationalise about 60 % of English data sets automatically (Dolmetsch 2000, p. 180). This seems to be a high degree of automation, but reveals on the second view the necessity for further improvements, as almost every second data set requires manual interaction.

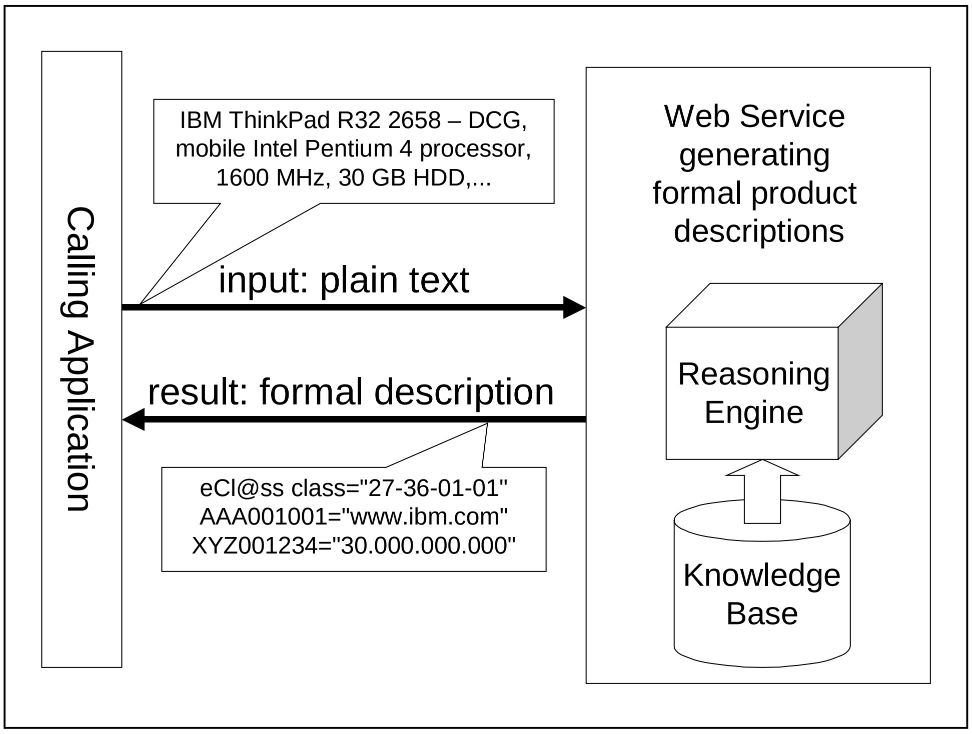
Calling Application
IBM ThinkPad R32 2658 – DCG, mobile Intel Pentium 4 processor, 1600 MHz, 30 GB HDD,...
input: plain text
result: formal description
eCl@ss class="27-36-01-01"
AAA001001="www.ibm.com"
XYZ001234="30.000.000.000"
Web Service generating formal product descriptions
Reasoning Engine
Knowledge Base

Figure 1. Deriving formal product descriptions from semi-structured input as a reasoning task.

Though catalogue management is the most obvious area of application for this functionality, there are many more. Any kind of higher level reasoning or agent-based decision regarding products first requires semantically encoded product descriptions. Thus, it makes sense to implement this functionality as an independent reasoning service. An appropriate method could look like the following.

public String formalProductDescription (String productDescription, String language)

The input parameter “productDescription” will usually be unstructured text data and the string parameter “language” should contain the language according to ISO-639. The resulting string will return a formal product description, e.g. the appropriate eCl@ss category plus values for all relevant attributes of that category. Figure 1 illustrates this kind of service.

2.3. Knowledge-based Size Conversion Service

Today’s garment sizes are no real standards, as size “extra large (XL)” from brand A might be smaller than “large (L)” from brand B. It is obviously impossible to order shoes just on the basis of the numerical shoe size, no matter which size system is being used. This is a major inhibitor for E-commerce in the clothing industry. A Web Service that reasons about suitable sizes (and, especially for shoes, suitable brands) would be extremely valuable and could foster the development of integrated shopping solutions for that segment. Such a service could be invoked in the following way.

public float determineShoeSize (String footShape, String manufacturer, float soleLength)

The input parameter “footShape” should be an XML document containing the foot shape as Scalable Vector Graphics (SVG) data and the string parameter “manufacturer” should contain the chosen shoe brand, either as URI (“www.lloyd-shoes.de”) or encoded as DUNS. The parameter “soleLength” represents the total sole length in inches or centimeters. Additional parameters (e.g. gender) might be useful.

The resulting float value will return the suitable shoe size, either in the EU or the US size system.

2.4. Further Examples

Another interesting area of application is **reasoning about risks:** Some business transactions do not take place in an integrated, digital way at present, because one or both contract parties are afraid of fraudulent opponents. In the history of trade, merchants and consumers have developed sophisticated skills for judging the probability of a fraudulent opponent. This judging is nothing but a reasoning task: An attempt to calculate the probability of a bogus transaction on the basis of incomplete knowledge. One could argue, however, that a simple database with a credit record of all possible business parties would do the job. But this does not stand against a thorough analysis, because the risk of a transaction might not be completely determined (and, if determined, not to be entirely derived from) the opponent’s identity. It might be necessary to know the ordered good categories, the payment details, or the delivery address.

public float determineFraudRisk (String customerAddress, String deliveryAddress, String orderedGoods, float total, String paymentDetails)

with the string parameters containing either unstructured text data or well-formed XML-documents. The resulting float value represents the probability of a fraudulent transaction (0=0 %, 1=100%).

More sophisticated reasoning services could generate statements on

- the amount of duties due for a given product description and the respective countries of destination and origin or
- transportation requirements.

3. Web Services for Reasoning Tasks

Such reasoning engines could, of course, be part of the internal logic of appropriate E-business applications. However, there are several arguments in favour of implementing them as independent components, e.g.

- better reusability,
- incentive conflicts,
- decoupling of the dynamics of the domains involved, and

- evolutionary improvement, facilitated through diversification and competition.

There is a general trend to split the functionality of solutions into parts, which arises e.g. from the “volatile nature of business requirements” and the “need to rapidly introduce new products in value chains” (Arsanjani 2002, p. 31). For methods that need to be remotely invoked in a distributed environment, Web Services seem to become the architectural component of choice. It is thus reasonable to implement the respective reasoning engines as SOAP-based Web Services.

3.1. Advantages of Self-contained, Encapsulated Services

One of the advantages of Web Services is that they are platform and programming language neutral (IBM Web Services Architecture team 2000, p. 1). They further help to avoid the brittleness of monolithic applications (IBM Web Services Architecture team 2000, p. 2) and make the resulting architecture less sensitive to environmental change.

3.2. Incentive Conflicts

With regard to reasoning services, economical aspects like trust, transparency, and incentive structures are equally important as technical details. Product comparison, for example, concerns the fundamental functionality of a market. It is obvious that this may lead to incentive conflicts. If the reasoning functionality is part of a monolithic software application, it will be very difficult for the users to detect biased reasoning results.

Self-contained Web services, of course, do not automatically solve all trust and incentive issues. They facilitate, however, comparison and selection by the users.

It is even possible that “meta” Web Services will evolve that judge the quality and trustworthiness of other Web Services.

3.3. Decoupling

If parts of a systems are subject to different domain dynamics, this should be considered when decomposing the functionality into a set of Web Services (Stiemerling 2002, p. 443). Otherwise, the diverse dynamics will overlay and maintenance will be extremely difficult. Imagine if an improved matching algorithm for the comparison of DVD players or an updated ontology for digital cameras required a release change of the whole catalogue data management solution.

This is consistent with the prevailing assembly-oriented view of software engineering, having modularisation and separation of concerns as key elements (Arsanjani 2002, p. 31).

3.4. Competition and Evolution

It will foster the evolutionary improvement of reasoning services if users and applications can easily switch between service providers. This is for two reasons. First, there will be competition among the providers. Second and more important, it will be a lot easier to compare two or more services with similar functionality. In the long run, this will yield better reasoning services. A third advantage will be the availability of backup services.

Thus, there can and should be multiple web services from different providers delivering similar functionality. This transforms the idea of second sourcing strategies from traditional industries to modular E-business applications. As Web Services can be described in a formal, machine-understandable way (IBM Web Services Architecture team 2000, p. 4), an application can automatically try to locate an alternative web service if the preferred provider is temporarily unavailable.

4. Implementation

Implementing Web Services that provide reasoning functionality poses several design issues, especially

- the design of a suitable reasoning engine and knowledge representation,
- internal and external knowledge representation and maintenance,
- ontological aspects (regarding the communication between caller and service on the one hand and the use of external sources of knowledge through the service on the other hand) and
- security, trust, and billing.

A prototype for product comparison is being developed at present. The further research will show to what extent these problems will be managable with reasonable effort.

5. Conclusion

Future E-business architectures will require access to reasoning services. Implementing them as self-contained Web Services will be the architecture of choice, because it solves a multitude of problems, of which incentive conflicts, decoupling of the domain dynamics, and the need for diversity are the most important ones.

I would like to thank P. R. Balasubramanian from the Boston University for valuable discussions and the School of Management at the Boston University for generous access to their resources.

References

Arsanjani, Ali (Arsanjani 2002): Developing and Integrating Enterprise Components and Services, in: Communications of the ACM, 45 (2002), 10, p. 31-34.

Dolmetsch, Ralph (Dolmetsch 2000): eProcurement. Sparpotential im Einkauf, München, 2000.

Glushko, Robert J.; Tenenbaum, Jay M.; Meltzer, Bart (Glushko/Tenenbaum/Meltzer 1999): An XML Framework for Agent-based E-commerce, in: Communications of the ACM, 42 (1999), 3, p. 106-114.

Greiner, Russell; Darken, Christian; Santoso, N. Iwan (Greiner/Darken/Santoso 2001): Efficient Reasoning, in: ACM Computing Surveys, 33 (2001), 1, p. 1-30.

Hepp, Martin (Hepp 2002): Interoperabilität, Metamarktplätze und agentenbasierte Arbitrageure, in: Dangelmaier, W.; Emmrich, A.; Kaschula, D. (Eds.), Modelle im E-Business, Paderborn, 2002, p. 475-489.

IBM Web Services Architecture team (IBM Web Services Architecture team 2000): Web Services architecture overview. The next stage of evolution for e-business, in: www-106.ibm.com/developerworks/web/library/w-ovr.

Stiemerling, Oliver (Stiemerling, Oliver 2002): Web-Services als Basis für evolvierbare Softwaresysteme, in: Wirtschaftsinformatik, 44 (2002), 5, p. 435-445.