The Knowledge Bazaar

Brian Craker[1] and Frans Coenen[2]
[1] Becoms Ltd, 69 Little Woodcote, Carshalton, Surrey SM5 4DD
Email: b.craker@knowledgebazaar.org
[2] Department of Computer Science, The University of Liverpool,
Liverpool, L69 3BX. Email: frans@csc.liv.ac.uk

Abstract

The concept of the Knowledge Bazaar as a paradigm for the development of Expert Systems, whereby knowledge bases are created dynamically using knowledge supplied by self appointed Internet communities is proposed. The idea espouses the creation of individual Knowledge Bazaars, operating in specific domains, but all operating through a generic Knowledge Bazaar XML Web application. Issues addressed include the provision of the service, XML rule representations and rule integrity. The concept is illustrated with a demonstration gardening Knowledge Bazaar that is currently operational.
Keywords: Knowledge Bazaar, WWW Rule Based Systems, XML.

1. Introduction

In this paper we describe an application of expert system technology founded on the idea of, what we have called, the *knowledge bazaar* (as opposed to a more traditional *cathedral* approach). The terms bazaar and cathedral (popularised in Raymond 99) are used here to distinguish between the traditional centralised approach to software development and an alternative, de-centralised, approach facilitated by the Internet. Use of the Internet to permit access to Expert System technology is not new, there are many examples. However, these all operate in a limited and very different manner to the Knowledge Bazaar concept as proposed here, in that they only allow users to pose queries. There are a number of reasons for the current limitations on the use of Expert Systems across the Internet, which are mostly concerned with security and (to a lesser extent) transmission speed.

The philosophical underpinning behind the knowledge bazaar is the observation that knowledge can be accumulated, not from a limited number of experts or expert sources, but dynamically from Internet users as they solve problems and offer advice. Consequently expert systems developed using the bazaar approach will be able to evolve. The knowledge contained in such expert systems might then be considered to be akin to the shared knowledge found in a (market) bazaar --- another reason for the use of the term.

It is suggested in this paper that the Knowledge Bazaar concept is an efficient, effective and immensely powerful way of harnessing the combined knowledge of

global communities of Internet users to develop and maintain expert systems. To illustrate the idea the authors have developed a generic Knowledge Bazaar XML Web Service. This generic Knowledge Bazaar facilitates communication and interaction with particular Knowledge Bazaars which, like traditional Expert Systems, operate in a specific domain (e.g. law, medicine, etc.). The Knowledge Bazaar communication model is illustrated in Figure 1. Note that, in the current demonstration system both the generic Knowledge bazaar and all specific Knowledge Bazaars are hosted on a single server. A gardening Knowledge Bazaar[1] has also been developed to illustrate both the principle and the operation of the Knowledge Bazaar concept.

The generic Knowledge Bazaar provides the interface to allow Bazaar users to submit queries to specific Bazaars, which are either:

Answered immediately if the answer is available in the system's knowledge base, or (If the answer is not available) posted to await an answer from members of a self appointed, on-line, community with respect to the domain.

In the second case, when an answer is provided, the Bazaar will update its knowledge base and post the answer to the user who originally posted the query. In this manner the knowledge (expertise) contained in individual Knowledge Bazaars will evolve with time.

In the remainder of this paper the background to the Knowledge bazaar concept is presented in further detail in Section 2. Design considerations are discussed extensively in Section 3 which includes much consideration of the available technology. In sub-sections 3.1, 3.2, 3.3 and 3.4 special consideration is given to: service-client communication, implementation of the Knowledge bazaar web service, the adopted XML rule representation, and rule integrity. The operation of the generic Knowledge Bazaar is considered in further detail in Section 4, and that of the demonstration gardening Knowledge Bazaar in section 5. The overall approach is evaluated in Section 6, and some final conclusions drawn in Section 7.

2. Background

The terms bazaar and cathedral in the context of software development were first popularised by Raymond. Raymond describes the cathedral approach as the traditional "monolithic, highly planned, top-down style" of software development; while the Bazaar approach, by contrast, involves a "chaotic, evolutionary, market-driven model" (Raymond 99). The Bazaar approach is evident in the open-source software movement e.g. the development of the Linux Operating System. Advocates of the Bazaar approach argue that it is more cost effective and produces a higher quality product than traditional "cathedral" type developments. Critics (for example Bezroukov 99) suggest that things are much more complex, i.e. it is quality rather than quantity that is important. However, Bezroukov does

[1] Available at www.knowledgebazaar.org

acknowledge that by removing geographic boundaries the Internet increases the quality of the pool of expertise. The Knowledge Bazaar is thus the application of Raymond's ideas on Bazaar development to knowledge gathering for Expert Systems.

Expert Systems have had a presence on the WWW for many years. Grove discusses a number of these (Grove 2000) --- one example is Acquired Intelligences' "Whale Watcher"[2]. As noted in the introduction, what most of these systems have in common is that the interaction is limited to querying. In the case of Whale Watcher the user is simply taken down a decision tree structure using a sequence of queries. Most of the current Expert Systems accessible over the internet tend to be very small scale (Adams 2001). It is suggested here that the Knowledge Bazaar approach will serve to significantly improve on the current Expert System presence on the WWW.

Many of the current WWW Expert Systems are written using the JESS (Java Expert System shell) rule engine which is designed to easily integrate with Java applications, which in turn makes JESS well suited to integrating Expert Systems with Internet applications. The XML markup language has also facilitated the provision of Expert System style WWW services. For example agent based systems that: extract rules from HTML pages (Shan 2003), or exchange rules between knowledge bases (Sedbrook 1998). Unsurprisingly XML has also been used to represent rules; in this respect it is argued that XML offers advantages of "interoperability, editability and searchability" (Friedman-Hill 2003).

There is also a significant amount of current research directed at the generic representations of knowledge on the WWW. Given the above the Knowledge Bazaar concept has been implemented as a XML WWW service using JESS as the expert system shell.

3. Design Considerations

The operation of any specific Knowledge Bazaar is facilitated through a Generic Knowledge Bazaar XML WWW Service. This allows remote users to interact with a domain specific Knowledge Bazaar (see Figure 1). Currently the Knowledge Bazaar system is implemented as a basic XHTML Internet site, with all interaction facilitated using forms (thus avoiding the need for natural language processing).

The XML WWW service is based on SOAP (Simple Object Access Protocol) with J2EE used to handle server side client support. The programming language used to implement the generic web service is, of course, independent of that used for individual Knowledge Bazaars, however since JESS was used as the Expert System shell it made sense to use Java for the WWW service.

---

[2] Available at http://www.aiinc.ca/demos/whale.shtml

The Interface to the web service is defined using the Web Service Definition Language (WSDL). WSDL is one of the essential building blocks for Web Services (Schmelzer et al. 2002). It is an official World Wide Web Consortium (W3C) standard which defines an XML grammar for Web Service definition. It describes both the operations of a Web Service and the format of the messages that are sent and received by it. A client uses the WSDL document to determine how to invoke the Web Service. WSDL provides a hierarchical definition. At the top level, the service is broken down into a number of port definitions. Each port represents the availability of a particular binding at a particular web address or endpoint. A binding corresponds to the implementation of a port using a specific protocol. Although SOAP was used to implement the Knowledge Bazaar service, the WSDL structure also allow for other protocols e.g. CORBA.

It is also worth noting here that the WSDL document could be published in a UDDI registry to allow for automatic service discovery. For the demonstration system described here this was deemed unnecessary. The WSDL document does however form the basis for the development of both the service and client parts of the prototype.

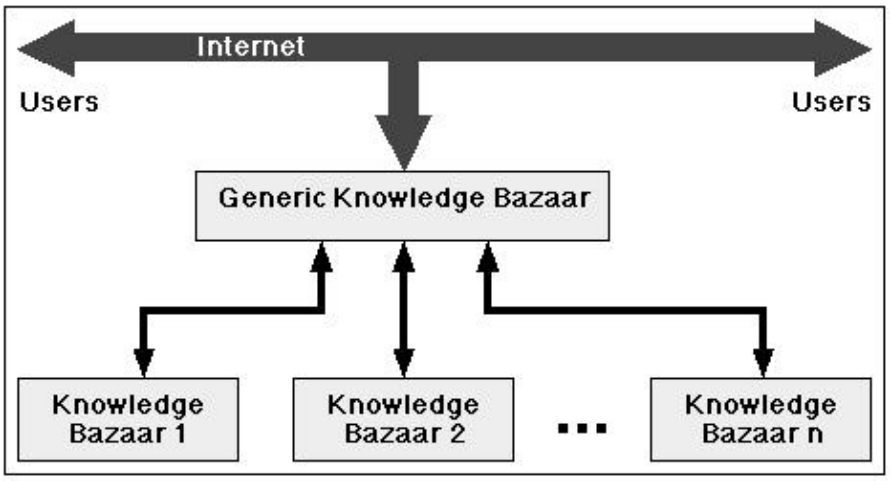

Figure 1: The Knowledge Bazaar communications model

3.1. Service-client communications

As noted above the WSDL document describes the interface in terms of XML. In the context of the Knowledge Bazaar concept the WSDL document declares that the communication between service and client should use the Simple Object Access Protocol (SOAP). This Remote Procedure Call (RPC) protocol is also an XML text based representation – each message consists of an XML document. This contains Envelope, Header and Body elements based upon the information described in the WSDL. The use of open text based protocols has many advantages for Web Services. For example they allow interoperability irrespective of whether

different technologies are used at either end, and they allow easy passage through firewalls.

For both client and server it was necessary to handle the transmission protocol for the SOAP messages and convert the contents from the XML message structures to Java type representations. It was also necessary to ensure that messages were converted into function calls and any "fault messages" were converted into exceptions. There are a number of possible approaches by which this may be achieved, with different vendors producing different SOAP interfacing packages. For example the Web Service Development Pack (WSDP) package provided by Sun, includes the SOAP with Attachments API for Java (SAAJ) that provides the "javax.xml.soap" Java package that in turn allows SOAP messages to be constructed directly.

An alternative to SAAI, and arguably a better choice for simple applications such as the Knowledge Bazaar concept, is to use an implementation of the Java API for XML-based Remote Procedure Calls (or JAX-RPC). The JAX-RPC API hides the complexity of the underlying calls to SOAP. There is no need to generate or parse SOAP messages and the JAX-RPC runtime system handles the translation between SOAP messages and API calls. JAX-RPC is a common standard, but there are different implementations available such as: Glue, Axis and the Sun WSDP implementation. There are also different ways to create a JAX-RPC application, but all involve using the WSDL file to generate files which perform the necessary translations between XML and Java.

For the service endpoint, the process involves running the WSCompile tool that forms part of an JAX-RPC implementation, during the software development phase (via the J2EE Application Server GUI or the command line). This tool processes the WSDL file and generates the interface and related Java classes associated with the complex types used in the interface. Skeleton interface implementation classes are also generated for each port endpoint. The developer then needs to flesh out the skeleton files with the functionality that the service should implement. When the Web Service is deployed additional JAX-RPC files are generated for the runtime environment.

For the client side of the interface, the link between the WSDL file and code generation is not so clear. Since the WSDL file is under the control of the Service provider, it may not be possible to use it directly. It is also necessary to acknowledge that the file could change without notice. Using a tool to generate static stubs would place an over-reliance on implementation specific classes. To address these concerns different methods for producing Web Service Clients have been developed. Generated files are obviously needed at run-time, however the point in time at which the interfaces to the Java code are generated can change.

For simple applications where the service and client are produced by the same organisation and the WSDL file is relatively static, the recommended solution is for the tool to generate static stubs offline and use these to access the service. The two alternative approaches are to use a Dynamic Proxy or a Dynamic Invocation Interface (DII). For Dynamic Proxy, the client makes the RPC call through a class

that is created at runtime. The client code does not rely on an implementation-specific class but the WSCompile tool is still required. For DII, there is no need to use an offline tool. All necessary files are generated at runtime directly from the WSDL file, rather than at the service endpoint. A client can therefore make a call even if the signature of the remote procedure is unknown at compilation time. For the demonstration gardening Knowledge bazaar it was decided to use the more powerful DII dynamic approach.

The chosen J2EE platform proved to be far from robust (e.g. incorrect generation of interface files if the name of an operation starts with an uppercase letter in the WSDL file --- although this should be perfectly acceptable). The level of support for XML schema types (e.g. string length restrictions) was also very limited, leading to the need to simplify the interface. "Workaround" solutions were found and implemented. However, with hindsight, the authors suggest that an alternative JAX-RPC implementations from a different organisations might have been better.

3.2. The Knowledge Bazaar XML Web Service Implementation

From the above the generic Knowledge Bazaar service was implemented using the JAX-RPC files generated from the WSDL. The additional functionality required for the service was provided by a combination of Java software and calls to an executing instance of the JESS Expert System Shell. The initialisation of JESS is performed using a batch file which creates structures, queries and local subroutines using the JESS language.

The Knowledge bazaar service is in effect half implemented in Java and half in JESS. JESS provides a very flexible interface. It is possible for the developer to choose where best to implement any routine – either internally within JESS or externally in Java using low level calls to JESS. Implementation within JESS is slightly less efficient, since subroutine calls need to be parsed. JESS is implemented in Java so it is more efficient if the JESS Java API is called directly from Java wherever possible. One advantage of using JESS code is however that it is "thread-safe". The allocation of user IDs is therefore best performed within JESS functions so that there is no chance that two users will be assigned the same value.

3.3. XML Rule Representation

Previous work has established that representing knowledge rules using XML has advantages in terms of 'interoperability, editability and searchability' (Friedman-Hill 2003); against the disadvantage of larger storage requirement. A number of projects are currently attempting to define standard, XML based, domain independent rule languages - for example the RuleML project (Wagner et al. 2004). It is clear that there are significant interoperability advantages from the development of an industrial standard in this area. These projects however are very general and still at an early stage of progress. Whatever the case only a small subset of such a standard would be applicable to a knowledge bazaar application.

With respect to the work described here a very simple XML structure, sufficient to implement the Knowledge Bazaar concept, was developed. Of course, if one of the current XML rule languages does develop into an industry standard, it will be easy to later transform rules generated within a Knowledge Bazaar into a RuleML (or another) structure using the XSLT XML conversion language. With respect to the Knowledge Bazaar system a very simple binary structure, which would allow simple object-operation-property tuples (propositions) to be expressed, was considered to be a sufficient representation. The syntax is presented in Table 1.

```
<Kbrule>
      <Object> … </Object>
      <Operation> … </Operation>
      <Property> … </Property>
</Kbrule>
```

Table 1: Knowledge Bazaar XML Rule Structure

Thus to express the fact that a Cox is a variety of apple, the following XML would be used:

```
<Kbrule>
      <Object>Cox</Object>
      <Operation>is_a_variety_of</Operation>
      <Property>Apple</Property>
</Kbrule>
```

(See Section 5 for further detail concerning the above example rule.)

The XML structure given in Table 1 allowed Knowledge Bazaars to effectively represent most simple facts. However, to take a greater advantage of the reasoning power of Expert Systems, it was felt necessary to also be able to express relationships between facts. Using the above structure this can be achieved by allowing the object and property elements to represent facts rather than nodes. The operation element can then be used to express the relationship between two facts. The gardening Knowledge Bazaar demonstrator therefore includes two special operations: `IMPLIES` and `NOT IMPLIES`. When the generic knowledge bazaar receives rule data with these operation values it will treat the object and property values as encoded facts rather than atoms and insert appropriate rules in to the Expert System.

The above also allowed the authors to keep the interface, between client and server, simple. It is anticipated that, with respect to potential future Knowledge Bazaar applications, a more complex grammar may very well be required to allow more complex rules to be expressed.

3.4. Rule Integrity

An important feature of the system is that it ensures both security and integrity of the knowledge contained in individual bazaars. If a user inadvertently, mistakenly or maliciously enters false information this is removed quickly. At the same time the system ensures that trusted information is harder to remove, and guards against inadvertent or malicious removal.

It is acknowledge that, no matter how well accepted information is now, it can become redundant through the passage of time. For some knowledge domains, e.g. IT support, the rate of obsolescence can be high. For the gardening Knowledge Bazaar demonstrator, most information is expected to remain constant; although factors such as new discoveries, plant breeding advances, global warming etc. may at some point mean that even here previously correct information might need to be replaced. Since the authors wished to fully adhere to the Bazaar principles for all aspects of the system, it was decided to use feedback from peer review, to guide the automatic integrity maintenance mechanism built into the system. To this end, users of the Knowledge Bazaars are encouraged to provide feedback as to whether previously supplied advice had proved accurate or false via a form driven interface. It is equally important that good as well as negative feedback is supplied. Currently only (trusted) registered users are permitted to contribute knowledge to the bazaar (though everyone can post queries). Associated with every user is a quality metric which is maintained by the system (not unlike the mechanism operated with e-bay). In addition there is a quality metric associated with every piece of knowledge contained within the system. The initial quality associated with the knowledge is based upon the current quality value of the user who supplies the information.

As users report feedback (or supply knowledge which conflicts or agrees), these quality values are adjusted. Negative feedback causes the quality of a rule to diminish, whereas positive feedback causes it to increase. If the quality of a rule diminishes below a threshold then it will be removed from the system and the quality of the original contributor reduced. Conversely, users that contribute information that is reported as correct by other users (not themselves) have their quality values increased. This process is illustrated shown in the Figure 2.

To date it has been found that the above simple mechanism ensures that errors can be corrected, but that when knowledge has proved useful to many people, it becomes harder to remove - so reducing the possibility of abuse. Information supplied by users who have contributed lots of useful information in the past is, at least initially, harder to remove than that supplied by new users or poorly performing contributors.

4. Generic Knowledge Bazaar Operation

The generic, knowledge domain independent, knowledge bazaar offers facilities for users to contribute knowledge either on their own initiative (unsolicited) or in

response to requests from other users (solicited). Users can also request advice or provide feedback as to the validity of content. The generic Knowledge Bazaar services these requests through communication with the appropriate domain specific Knowledge Bazaar (as illustrated in Figure 1).

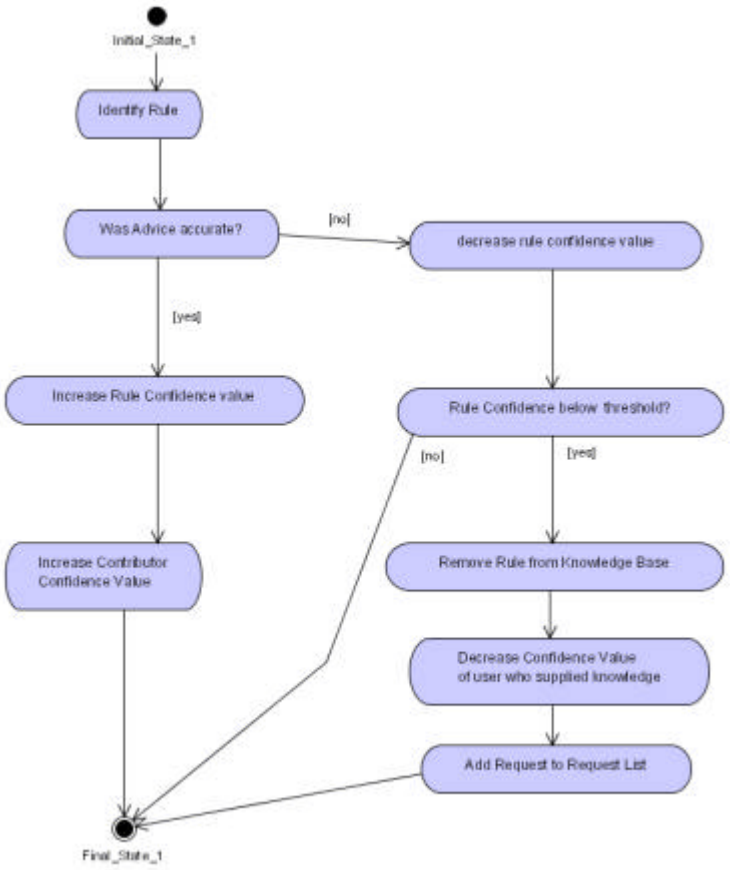

Figure 2: Knowledge Bazaar Rule Integrity Mechanism

In the case of the Generic Bazaar Service, three ports were identified as useful for the service:

- A User Data port (to handle user registration and logon verification).
- A Knowledge Bazaar port (to handle the Knowledge Bazaar processing for normal users).
- A Supervisory port (to allow the client operator to access performance data and to perform maintenance operations).

(Note that a client could choose to handle user registration locally or may have no need for the supervisory functionality. However the port separation ensures that the core bazaar functionality is isolated and clearly identified.)

The generic knowledge bazaar interface has been designed using industry standard Web Service technologies which ensure the development of a client is relatively straightforward. WSDL is used to unambiguously describe the service interface. The service would be published using a publicly accessible UDDI directory service. An interested party would be directed from this directory service to the location of the WSDL file. This file would then be processed by the tools to generate the language specific (e.g. Java) interface as described in section 3.1. The generic service provides the basic knowledge handling functionality. A client developer then just needs to develop the presentation of the functionality to suit the specific application. The client development task does not require any knowledge of expert systems. It just requires standard web site development skills.

5. Demonstration Gardening Knowledge Bazaar Operation

For the demonstration gardening Knowledge Bazaar users interact with the client application using a form-driven WWW interface. Users contribute to the system either on their own initiative or in response to requests from other users. If someone provides an answer to a request then the interested party is informed and the expert system updated. If the same question is asked again, the answer will be obtained directly from the expert system. The expert system thus continually expands its knowledge base - without needing to use the services of a traditional expert.

Individual Knowledge Bazaar clients use a form driven interface that will restrict the possible entries for operations and properties (corresponding to the rule representation described in section 3.1). The object field will allow unrestricted text entry. These restrictions avoid natural language processing complications. With respect to the gardening knowledge bazaar Table 2 shows the values that are permitted:

For queries, the user will be asked to supply an Object and an Operation and the associated Property will be sought.

For knowledge contribution, a simple interface allows users to enter facts in the form of object-operation-property tuples. It is anticipated that in the future a more complex interface will allow a user to specify two complete facts (as object, operation, property tuples) and state if one fact implies, or does not imply, the other fact.

For Knowledge Bazaar applications, it might be that some form of Fat Client Peer to Peer architecture could offer some advantages. If instead of storing all

knowledge centrally it was distributed across many machines then it might be possible to mitigate the storage costs and processing delays that could arise for large knowledge bases. However, in the interest of ease of access and simplicity of design, the authors opted for a thin client application as the most appropriate choice.

The demonstration gardening Knowledge Bazaar is currently available at http://www.knowledgebazaar.org, interested readers are invited to interact with the system to obtain a better appreciation of its operation.

Operation	**Property**
`Is a variety of`	No restriction
`Flowers in`	Summer, spring, winter,
`Has longevity`	Annual, perennial, biannual
`Has fragrance`	Low, medium, high
`Has sun requirement`	Low, medium, high, easy
`Has temperature requirement`	Low, medium, high, easy
`Has frost tolerance`	Low, medium, high
`Has water requirement`	Low, medium, high
`Has soil ph requirement`	Acid, alkaline, easy

Table 2: Gardening Knowledge Bazaar Rule Restrictions

6. Evaluation

At time of writing (May 2005) the gardening Knowledge bazaar demonstrator had been in full operation for several months only. Feedback from a survey conducted by the company, where some fifty participants were asked to interact with the demonstrator, indicates that the basic Knowledge Bazaar concept has been well received. The majority of respondents gave a very positive reaction to the knowledge bazaar concept. Over half (i.e. more than 25 of the respondents) agreed that they would use such a system as a source of information with only 3% disagreeing (the rest were undecided). Even more people (65%) said they would be prepared to contribute their knowledge to such a system with only 6% disagreeing. The basic Knowledge Bazaar concept is further supported by the observation that online communities have proliferated over the last few years, indicating that Internet users are happy to provide answers to problems/questions (for example using “message boards”).

With respect to rule integrity half of the respondents believed that Peer review would be sufficient to ensure the quality of the knowledge base. The reaction to the form driven interface designed to support Knowledge Bazaar applications was also interesting in that most respondents regarded a Natural Language Processing (NLP) interface as a low-priority enhancement rather than a necessity.

The authors always considered the threat of abuse to be the greatest challenge to the Knowledge Bazaar concept in that the soliciting of knowledge would allow users to post objectionable content as queries. The Survey participants were asked for their views on the best way to tackle this. Most (47%) thought that some kind of filter mechanism should be introduced despite the time and cost implications. Peer review was however thought to be sufficient by a large minority (35%). The option of requiring personal details for membership was less popular (18%), reflecting the objections that many people also have to registering with "message board" Internet sites.

The positive results obtained from the survey were further supported by the authors' intuition that experts systems developed using the Knowledge Bazaar approach more accurately reflect the way in which knowledge is applied. The authors believe that the traditional approach to building Expert Systems, which assumes knowledge can be detached from its social context, is flawed. Even during the "golden age" of Expert Systems, Bobrow was already advocating the need for "Community knowledge bases that integrate expertise from many different sources" (Bobrow 1986). This view is supported by Wenger (1998) who argues that knowledge, and the process of learning, involves much more than information gathering or technology. Wenger suggests a number of components to learning that are embraced by *communities of practice*. Wenger's ideas about communities of practice with respect to knowledge elicitation strongly support the Knowledge Bazaar concept as proposed here.

7. Conclusions

In this paper the idea of the Knowledge Bazaar approach to building expert systems has been introduced. The term "Knowledge Bazaar" has been proposed to describe the concept of a body of knowledge that evolves dynamically using the contributions, supplied across the Internet, of self-selected individuals. To illustrate the concept a gardening Knowledge Bazaar demonstrator has been developed which has been well received. The feedback indicates that the concept has great potential for harnessing the knowledge available across the Internet so as to build genuinely useful knowledge based systems and applications.

References

1. Adams, J (2001). *The feasibility of distributed web based expert systems.* Proc. IEEE Systems, Man, and Cybernetics Conference.
2. Bezroukov, N. (1999). A Second Look at the *Cathedral and* the Bazaar. First Monday Journal, Vol 4, num 12 (Available from http://firstmonday.org/issues/issue4_12/bezroukov/index.html).
3. Bobrow et al (1986). *Expert Systems: Peril and Promise*. Communications of the ACM. Vol 29, Num 9.

4. Friedman-Hill, E (2003). *JESS in Action – Rule-Based Systems in Java.* Manning Publications.
5. Grove, R. (2000). *Internet Based Expert Systems.* Expert Systems, Vol. 17 No.3.
6. Raymond, E. (1999). The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly.
7. Schmelzer, R., Vandersypen, T., Bloomberg, J., Siddalingaiah, M., Hunting, S., Qualls, M., Darby, C., Houlding, D. and Kennedy, D. (2002). *XML and Web Services Unleashed.* SAMS Publishing.
8. Sedbrook, T (2001). Integrating e-business XML business forms and rule-based agent technologies. Expert Systems, Vol.18, No.5.
9. Shan, F et al (2003). A programmable agent for knowledge discovery on the Web. Expert Systems, Vol. 20, No. 2.
10. Wagner, G., Antoniou, G., Tabet, S. and Boley, H. (2004). *The Abstract Syntax of RuleML - Towards a General Web Rule Language Framework.* Proc. Web Intelligence 2004: 628-631
11. Wenger, E. (1998). Communities of Practice – Learning, Meaning and Identity. Cambridge University Press.