Architecture-Based Integration of CBR-Components into KM-Systems

Norbert Gronau[1], Frank Laskowski[2]

[1] Universität Oldenburg, Abteilung Wirtschaftsinformatik,
Escherweg 2, 26121 Oldenburg, Germany
Norbert.Gronau@informatik.uni-oldenburg.de
http://www-wi.informatik.uni-oldenburg.de
[2] OFFIS e.V., Escherweg 2, 26121 Oldenburg, Germany
Frank.Laskowski@domino-wi.offis.uni-oldenburg.de
http://www.offis.de

Abstract. Knowledge-Management (KM)-systems integrate multiple technologies with the purpose of supporting users who have "knowledge-related" problems. Case-Based Reasoning (CBR)-technology is well known for improving problem solutions. Combining both concepts the project "TO_KNOW" has the goal to design a CBR-component that is adaptable to typical KM-Systems. The basic idea is to take into account user-queries as (intermediate) problem-solutions for getting the needed information from the KM-System's discovery services.
In this paper we describe an architecture-based approach to design the CBR-component yielding a light-weighted integration into a KM-system. This is accomplished by looking at both parts designated for integration in terms of their architecture, i.e. the modularity of data and functionality. Since different KM-systems will allow various degrees of synergy, the process should result in a highly modularized CBR-component being ready for different models of integration. An overview about our design-method is given as well as examples covering first results on a more detailed level.

1 Introduction

Knowledge management-systems integrate a wide variety of method and technologies - from database-systems over CSCW-tools up to WWW-portals with the intention to support „knowledge-intensive“ processes. This especially indicates the use of knowledge oriented technology, like it’s well-known from the sector of artificial intelligence (AI).
Part of the project „TO_KNOW“ is to elaborate a proposition how KM-systems can be completed by case based reasoning. This article describes the starting point of the project, our method of proceeding, and sketches first results. In section 2 the idea of integrating CBR in KM-Systems is being motivated, together with a short repetition of underlying concepts. Section 3 introduces the architecture based approach used to design a CBR-component, which can be integrated into typical KM-systems. Section 4 lists different examples for the use of this approach and shows first results.

2 Case-based reasoning, knowledge management and the "TO_KNOW"-project

The aim in using KM-systems (e.g. [Ch98], [RS99], [KG00]) is to make electronically managed information available beyond operative and local purposes for institutions and enterprises. As „knowledge" this information should also be open to unforeseen and therefore more free creative reuse.
Figure 1 shows a KM-systems architecture this article relies on. The figure subdivides the KM specific functional sections and components roughly in terms of a layered architecture. (Further it is emphasized that from the technical point of view the open and established internet-standards should be used in the communication-tier.)

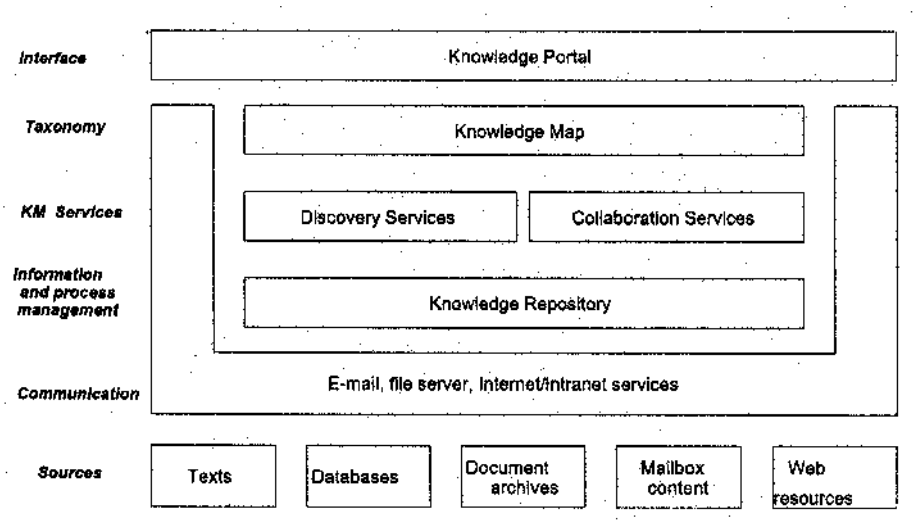


Fig 1. Architecture of a KM-system ([Ve99])

Components KM-systems are based on for this purpose are for example document- & content-management, CSCW-tools and portals. The technical achievements of these components (for example efficient handling of large amounts of data) are acknowledged and need to remain focused in research, if in the future KM-systems shall achieve more than just superficial (and therefore very limited) integration.
The integration of as many as possible heterogeneous data sources is one of the essential achievements of a KM-system. This especially includes not yet integrated, new sources. Therefore the technical format, internal structure and even the „domain" of these data sources cannot be fixed. In so far a KM-system cannot carry out a complete „knowledge modeling" in the sense of AI which would then be expected to semantically integrate new data source in a clairvoyant way.
Nevertheless there is need for interpretation techniques as already developed in the field of AI. That's because a KM-system is positioned in between the human user and data from the data sources with which the user is only roughly familiar regarding

contents and practically not acquainted at all regarding formal aspects. Subjectively the user acts not on „data“, otherwise he would be aware of storage locations, access software etc. and also semantic aspects like the name of attributes in structured data. Thus is appears that supporting the abstraction “from data to knowledge“ is a central achievement of a KM-system.
In this context the quality of Discovery Services of a KM-system is essential. Particularly the Information Retrieval (IR) (e.g. [Ri79], [IR01]) is a functional bottleneck because search-tools are probably the best known and most accepted means for users and because of the fact that other services, e.g. push-techniques, are based on IR-services. A lot of relevant information may be integrated in a KM system - if there is a poor “understanding” of search-requests the probably very good quality of available data wouldn’t reach the user. This is the starting point for the project „TO_KNOW “ of which the basic ideas will be presented in this article:
Case-Based-Reasoning (e.g. [AP94]) is one approach of AI which on one hand has the potential to improve the quality of an IR component in a KM system, especially concerning the interpretation of search-queries. On the other hand CBR has lots of structural similarities to IR, which are clues for both conceptual and technical integration:

- The basic methods are inductive: conclusions are made up from existing data.
- Similarity metrics play an important role and are applied to the same information.
- A large number of single requests is well supported.
- No domain representation is required, esp. no specific modeling of the knowledge base is necessary. Instead data being inferred from the knowledge base will be managed additionally.

In view of these parallels the question rises whether CBR is effectively “included” in IR. Are modern IR methods conceptually already able to draw those inductive conclusions that could be provided by a CBR-system to enrich the IR process? E.g. when a CBR-system adds a keyword to a search-query (the “case”) because - as experience has shown in previous cases- the new keyword is connected with one present in the query, so in a classical IR component the same completion can be effectively carried out using a thesaurus. - A theoretically backed up clarification of this issue is not intended by this article but two points practically motivate the intention of widening IR-components of KM-systems by CBR-methods:

- Those systems with KM-tasks being used in business practice, among them also dedicated KM-systems, often use full text search engines as IR components. Accordingly concepts are missing, such as adaptivity, automatic building of associations etc., which a CBR-component can introduce into the KM-system.
- A CBR component adds different conceptual approaches and therefore possibilities for controlling and “tuning” to an IR-component working with keywords and thesauri. This is of importance if e.g. a KM-system has to be established in a large organization and a respectively huge amount of data and search-queries.

Besides the fundamentals in the fields of KM, IR and CBR results in the area of Textual CBR (TCBR) (e.g. [WB99]) need to be considered, especially those concerned with the synergy of IR and CBR (e.g. [RD95]). Mainly TCBR focus lies on understanding queries expressed in natural language and on using IR-methods in CBR. Based in the area CBR/KM (cp. [AM99]) the "TO_KNOW" project centers on typical (i.e. esp. formally expressed) search-queries and intends to use CBR as a component in IR.
Considering these reflections the aim of the project „ TO_KNOW" is to propose a CBR component that is able to improve the Discovery Service of a KM-system:

- The effectiveness of IR in typical KM-systems, i.e. performance regarding „precision", "recall" etc., should be increased by the CBR-component.
- For that the adaptability of the CBR-component is a requirement. Therefore the typical architecture of a KM-system has to be taken into account. The component itself should not prescribe the architecture of an integrating KM-system and therefore limit its openness.
- Efficiency (in broader sense) of the Extension is also required. The CBR component should be able to use existing and accessible functionality of the integrating KM-system, instead of consuming resources as a technical heavy weight including its own DBMS, middleware-concept and web-server.

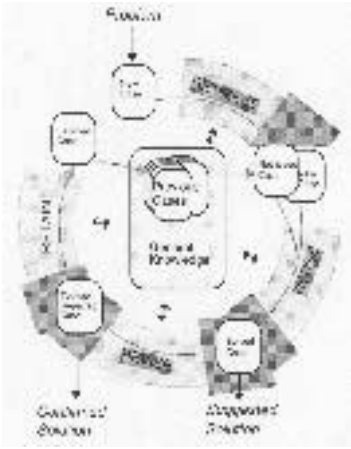
3 The architecture-based approach

To reach the development goals described above it makes sense to follow the aforementioned parallels of CBR- and IR-technology. This way procedures and aata of the components that are to be integrated can systematically be compared. In doing so it has to be checked if any parts or facets of CBR and IR have a correspondence of one of the following types:

- replace:
  Do data or tasks play a similar role as well in IR as in CBR?
- use:
  Can any aspect of the IR-component be used as a concrete technical part for the implementation of aspects of CBR-functionality?
- mapping
  Real KM-systems cannot be expected to be sufficiently open to support every meaningful replacement or use completely. In these cases a loose coupling should be possible, which is to be preferred over a complete „blindness" of the systems towards each other.
- add
  The notions of "replace(ment)", "use" and "mapping" describe interfaces between the CBR-component and the KM-system. Such are also to be specified for the exclusive parts, that represent the "core" of CBR- or IR-components.

- feedback
  Esp. CBR-specific data can be used in KM-systems – in the opposite direction to the "use"-correspondence. Thereby the CBR-component will provide some added value outside the extended IR-process.

The result of this analysis is an architecture that centers the design on interfaces and possible compositions. (cp. [Ga96], p. 18 for general advantages of this approach) This differs from [AP94] centering the problem of integration on the knowledge base. Anyhow [AP94] provides the CBR-cycle shown in figure 2 that is one of the two starting points of our analysis. It is compared to an corresponding IR-cycle (Figure 3).



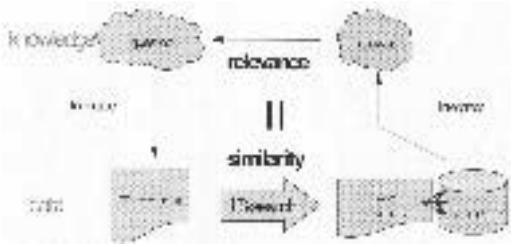


Fig. 2 & 3. CBR-Cycle ([AP94]) and IR-Cylce

4 Examples for the architecture-based integration

The following sections cover some concrete examples for applying the design process described in the last section. These examples illustrate basis points of comparison and outline early results of the project. The first section contains a description of the macro-architecture of the planed CBR-/KM-integration simultaneously providing an example for the correspondence-type "add". In the following sections examples are given for the "replace", "use", "feedback" and "mapping" types of correspondence.

4.1 The basic relationship: Adding the CBR-component to the KM-system

Objective of the "TO_KNOW"-project is to provide a component improving the IR-quality of a KM-system. Accordingly regarding the KM-system as well as the added CBR-component the use case to be considered is a user's search for information that is relevant to a given problem. A search-request represents thereby a problem description. The result of the corresponding search-query, i.e. a set of references to

documents form the knowledge base of the KM-system, can be considered a solution of the case. But we suppose to take into account some arguments not to establish this identification directly: The success of a search-process as a whole rather not depends on the concrete set of documents delivered to the user but on the "understanding" of the user's intentions. The resulting set of documents resulting provides only little indications whether the query has been well (or poorly) "understood". As well system-users realize their problem not as "sets of documents" but in terms of a search-query, its operators and syntax. (For a search-result reworked by a CBR-component by adding or replacing documents the improvement of quality cannot easily be explained: To state, that adding these 15 documents improves the quality of the result, does not clarify too much.) Moreover we assume that data in a KM-system tend to be more dynamic than its domain: A request that had been "well understood" - e.g. concerning the use of object-oriented technologies, picture-reportages in foreign countries or hermeneutic exegeses - results probably in another suitable set of documents when applied at a later date even if major numbers of documents that where part of the former result aren't anymore up to date, altered, or removed. To modify a newly computed result set by adding parts of an old one might even turn out to be counterproductive. This problem can esp. arise, in case the IR-component of the KM-system has been improved in a way not affecting the CBR-system that afterwards "reestablishes" old results with inferior quality. In this sense the design issue "request = case" discussed here exemplifies a characteristic feature of the projected CBR-component: It must not obstruct other components or future extensions of the KM-system. Hence if the document set collected beforehand by the IR-component is manipulated at all, the CBR-component should preserve the format of the result set to allow other components to add different functionality as well. So the CBR-component is corresponding to the KM-system by "adding" its "reuse"-phase to the IR-functionality. Figure 4 illustrates this type of correspondence.
Accordingly a search-request is being received by the KM-system as usual. (We omit changes to the user interface here, e.g. to support the "revise"-phase or to give control over the application of the CBR-method to the user.) The query is handed over to the CBR-component and eventually modified. The IR-component takes over to process the query. The CBR-component is being invoked a second time and finally the KM-system delivers the result to the user.

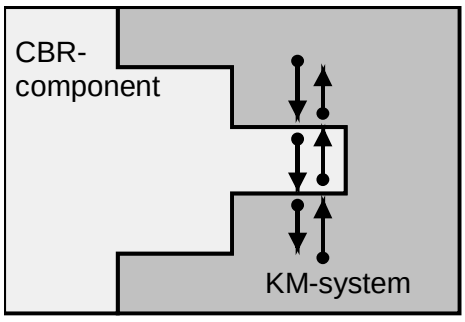


Fig. 4. Correspondence-type "add"

(Even when the result is not being manipulated, a second invocation of the CBR-component can be valuable to gather data to be used in the "revise"-phase, e.g. to identify cases with empty solutions or unreasonably large result sets.)

4.2 "Lean" integration (I): Using the Knowledge Repository to back-end the case-base

A CBR-component relies on a facility storing its case-base persistently and offering fast access to it. Technically spoken this facility has to be a "heavy" system-component just like the Knowledge Repository that is already part of a KM-system and can probably be accessed through an interface. This opportunity should be planned for by the CBR-component defining an inner interface to the case-base storage that thus can be implemented by a typical Knowledge Repository. If the KM-System does not support this optimization or if the costs to implement appropriate wrapper-software are judged too high by the operator of the KM-system, the CBR-component should provide its own storage facility. The correspondence between case-base and Knowledge Repository we classify as "use", because the Knowledge Repository stores the case data, but it is not expected to do so retaining the structure appropriate for interacting directly with the CBR-System in place of the case-base. The core-component is using a more abstract, CBR-oriented interface instead, that might also be implemented directly and hence efficiently by the CBR-system's own storage facility. Between this interface and the aforementioned one to access the Knowledge Repository there is the "using" part of the CBR-component. Schematically this is shown in figure 5.
On principle the CBR-component allows to use one of the data sources established in the KM-System instead of the Knowledge Repository. This may be favorable from a technical point of view, esp. if the data source is an efficient DBMS. Indeed therewith the CBR-component becomes directly dependent on the is-infrastructure outside the KM-system, and hence "individual software" is created.

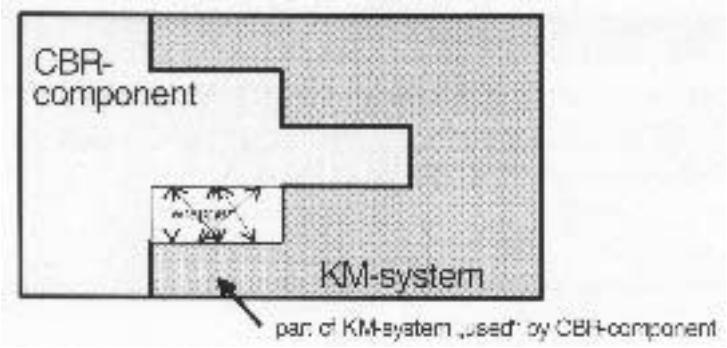


Fig. 5. Correspondence-type "use"

4.3 "Lean" integration (II): Replacing the CBR-system's similarity metrics with that of the KM-system

During the "retrieve"-phase a CBR-system searches for cases describing problems similar to the one that is to be solved currently. More concrete the CBR-component is looking for retained search-requests that match the current one. Indeed, to carry out this task is one of the essential capabilities of the available IR-component: to compare search-requests (and documents) with respect to similarity - as the case may be especially adopted to the domain of the KM-system. To outrival this feature the CBR-component would have to implement some major part of an IR-component redundantly or extend the KM-system by adding missing IR-techniques. Both kinds of extension should be done by improving or exchanging the IR-component itself. (Moreover the CBR-component should not be affected by such a procedure at all.)
Instead the match can apparently be done by the existing IR-component. Due to the significant role of this function and because of probably rather straight implementation we classify the matching-functionality of the CBR- and the IR-component as "replacing". Figure 6 illustrates this type of correspondence schematically.

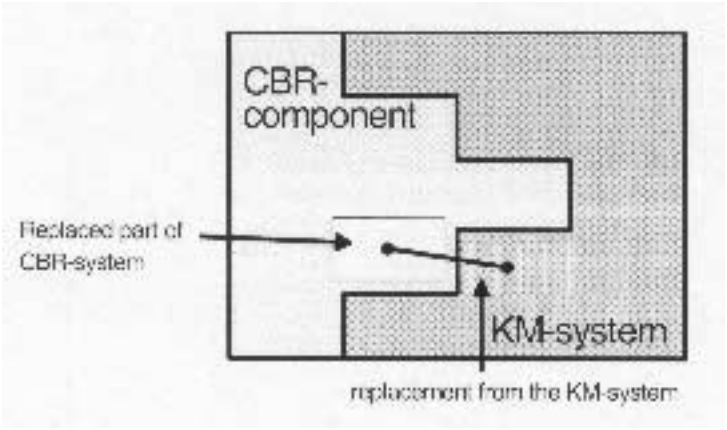


Fig. 6. Correspondence-type "replace"

Typically IR-components will offer the possibility to compare documents via an API. Thus the CBR-component should provide an interface to use this service replacing the according parts of the "retrieve"-phase. One naive approach is, to transform every case into a document for this purpose. To take advantage of the efficiency of the IR-component, the cases should be indexed as soon as they are retained in the KM-system: This might be implemented - as explained in the section above - by using the Knowledge Repository to store the case-base. Another possibility is to provide the case-base to the KM-system as an additional data source. Carrying this forward the simplest way is to store the search-requests from the case-base as documents. Given a new case to find matching cases during the "retrieve"-phase means just to issue

another IR-request - configured to make the IR-component only search for documents pertaining to the case-base.

4.4 Another kind of added-value: Feedback of the collected experiences

The main advantage of the CBR-component described in this article is the improvement of the KM-systems Discovery Services. Yet the CBR-component collects experiences that can be made available to the KM-system, too. We call this type of correspondence shown in figure 7 “feedback”. If e.g. a surpassing number of similar cases is evaluated to propose unsuitable solutions this might be an indication that according external data sources should be taped or that available appropriate documents are not classified and indexed as needed. Besides defining an interface to access the case-base from the KM-system we have to figure out which CBR-specific key data should be accumulated and held in the Knowledge Repository ready to be used by yet unknown applications that are to be integrated into the KM-system. Hypothetically a CBR-component with a large case-base might even contribute to maintaining the thesaurus used by the IR-component. However this kind of feedback would only be possible with the other involved components offering interfaces and access to their internal structure unlikely to be found esp. in commercial products.

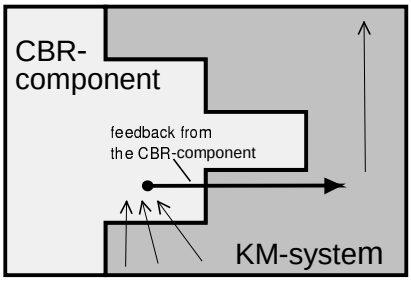


Fig. 7. Correspondence-type “feedback”

4.5 In difficult circumstances: Mapping user-profiles

Finally there is left the realistic scenario of an imperfect connection between the CBR-extension und the extended KM-system: Due to limited openness or insufficient efficiency coupling by “using” or “replacing” might be prohibited even though it is conceptually preferable. The KM-system could e.g. provide an interface allowing to retrieve the user-profiles but not offering the possibility to manipulate these. This constellation would block the CBR-component to add some of its accumulated experiences to the user-profiles maintained by the Knowledge Repository. If the response time of the Knowledge Repository is too long to handle a large amount of

requests in a satisfactorily efficient manner, a daily replication of the user-profiles is necessary. The CBR-component has to be able to store and synchronize such "extra-data". We have not yet found out, what data from the KM-system is candidate for this kind of "mapping" (see figure 8).

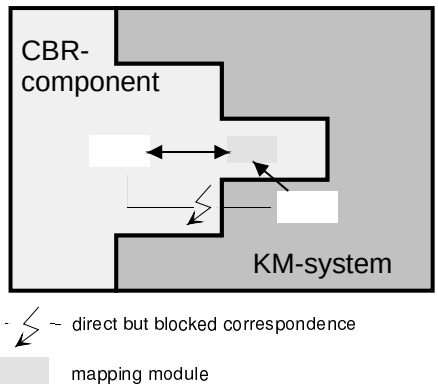


Fig. 8. Correspondence-type "mapping"

5 Conclusions

One of the goals of the "TO_KNOW" project is to design a CBR-component especially to extend typical KM-systems. More precisely the CBR-component shall extend the KM-system's Discovery-Services by handling an user's search-query as a case.

In this article we described our proposal for an architecture-based method to design an effective, integrated and efficient solution. This is achieved by systematically comparing an ideal CBR-system with a typical but fully-fledged KM-system. We call this process "architecture-based" because it consists of the comparison of modules from both systems and covers diverse levels of granularity. Results of this process that come out to be relevant for the design are grouped by their "correspondence-type": On the one hand modules from the CBR-component and the KM-system are corresponding by "replacing", "using" or "mapping" to one another – aiming at an efficient integration. On the other hand there are chances for the CBR-component to "add" its capabilities to the KM-system or to provide "feedback" thus improving the effectivity of the Discovery Services.

Subsequently examples are given to demonstrate the design-process and sketch some more concrete, early results of the project:

- How to use CBR-technology as part of the IR-process.
- Conceptual and technical opportunities to keep the CBR-component "light weighted".
- The idea to benefit from the integration beyond the "search-engine" by providing metadata as feedback from the CBR-component into the KM-system.

Planning the next steps we are aware of the potential of the current results in CBR-research to further adjust and improve the ongoing design process.

References

[AP94] Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. AI communications 7 (1994), 1, S. 35-39

[AM99] Aha, D.W.; Muñoz-Avila, H. (Hrsg.): Exploring Synergies of Knowledge Management and Case-Based Reasoning: Papers from the AAAI 1999 Workshop. Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, 1999

[Ch98] Choo,Chun wei: The Knowing Organization - How Organizations Use Information to Construct Meaning, Create Knowledge, and Make Decisions. Oxford University Press, 1998

[Ga96] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns, Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994

[IR01] Home-Page der ACM Special Interest Group on Information Retrieval Http://www.acm.org/sigir/, 3.12.2001

[KG00] Krallmann, H.; Gronau, N. (Hrsg.): Wettbewerbsvorteile durch Wissensmanagement - Methodik und Anwendungen des Knowledge Management. Stuttgart 2000

[RD95] Rissland, E.L.; Daniels, J.J.: Using CBR to Drive IR. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), 400-407, 1995

[Ri79] van Rijsbergen, C.J.: Information Retrieval. Second Edition, Butterworths, London, 1979

[RS99] Rigo, R.; Schönherr, M.: Wissensmanagement-Tools und -Dienstleistungen. Industrie Management 15 (1999) 6, S. 95-112

[Ve99] Versteegen, G.: Knowledge Management. Architektur für das Firmenwissen. i'X (1999) 3, S. 113-119

[WB99] Wilson, D.; Bradshaw, S.: CBR Textuality. In Proceedings of the Fourth UK Case-Based Reasoning Workshop, 1999