Considering the Dynamic in Knowledge Based Configuration

Ingo Kreuz[1]

Abstract. Learning from previous solutions could be a key for improving both the solution process and the quality of the solution. Unfortunately knowledge (i.e. components) in technical configuration domains changes quickly and learning brings with it a certain amount of conservatism. For example a component that was learned to be good for a certain task should no longer be chosen, if a newer version appears on the market. On the other hand a certain amount of conservatism is often desired since uncontrolled innovation is as a rule also detrimental, i.e. the newer component mentioned above is not obviously better.

This article presents Relevant Knowledge First (RKF) as a method and heuristic respectively. It tries to find a good compromise between conservatism and innovation based on statistical values and aging of knowledge.

1 INTRODUCTION

Technical markets including computers, multimedia or digital cameras change very quickly. Other technical domains use devices from these fast changing domains, such as cars, trains and airplanes. If one tries to put up a configuration system in these domains he or she will be confronted with the problem of fast changing facts in the knowledge bases. As a rule it is difficult to identify knowledge that is no longer necessary, so it is normally impossible to delete such knowledge. As a consequence the knowledge bases become larger and larger, thus slowing down the configuration processes.

Looking at the field of cognitive psychology we can get an idea of how human experts handle large amounts of quickly changing knowledge: We seem to be able to *concentrate* on actual tasks, we can *learn* from doing something successfully and repeatedly, and above all we are able to *forget* things that are no longer relevant – without deleting them.

In his book [1] Anderson calls the "concentration" the "*activation* of a memory trace". It indicates how accessible information is for a current problem, i.e. how fast and with what probability the information can be accessed. Every time we use a memory trace its accessibility increases a little. If a piece of information is needed frequently, its *action potential* increases what we called "*learn*ing" above. The relationship between the quantity of exercise and the access efficiency (e.g. measured as reaction time) results in a power function which is known in cognitive psychology terms as the *power law of learning*. The learning of information counteracts forgetting: If information is not used over a long period of time, its action potential becomes less. Experiments show that forgetting can also be described as a power function, which amongst other things could be explained by the decaying processes of the neural connections.

The concentration on a given task, the learning and forgetting have lead us to a method for assessing the *relevance* of information: RKF (Relevant Knowledge First) is a heuristic or method for the processing of knowledge bases in the field of configuration. It identifies the *relevance* of information for a given problem thereby simulating some kind of learning

- to speed up and to improve the solution process
- and the solution's quality

and forgetting

- to keep the search space small
- and to give "newer" knowledge the chance to prove its worth, avoiding conservatism.

Note: We do not want to simulate the human model described by Anderson. It simply served as a good starting point for the development of RKF. What we want is a measurement for the relevance of information in knowledge bases for a given task. Though the investigations of cognitive psychology gave us valuable ideas, we will leave this field now and introduce our measurement for relevance.

2 PRINCIPLE OF RKF

We recognized that it is useful to assess knowledge during a knowledge-based search process in order to be able to focus on an actual solution process. With this even large knowledge bases can be scanned efficiently. For this assessment there are two deciding factors which correspond to the antagonism between conservatism and innovation: On one hand the knowledge is very probably useful again if it has already been useful for similar tasks. On the other hand new information should be preferred, in order to obtain innovative solutions and avoid conservatism. For the assessment we use "relevance" which is calculated by age and usefulness of knowledge.

With RKF (Relevant Knowledge First) the search for solutions is supported by relevant knowledge being processed preferentially. When knowledge is selected during a solution process, e.g. in order to bring an object into the solution set or to check its consistency, the relevance for all knowledge *in question* is calculated and *one of*

[1] DaimlerChrysler AG, Research and Technology, HPC T721, D-70546 Stuttgart, Germany, email: ingo.kreuz@daimlerchrysler.com

the most relevant objects is used. For subsequent search processes the relevance is increased for successfully used information.

In order to avoid premature convergence on "bad" results, *the* most relevant knowledge is *not* always used. Instead of this only "one of the most relevant" pieces used. This can be achieved with the help of a random generator, whereby the probability for one choice should be proportional to the relevance of the knowledge.

The relevance of a piece of information i is calculated as functions of its age $a_{i,tc}$ (forget) and its usefulness $u_{i,tc}$ (train) for a given task class tc, whereby usefulness compensates for age. The constant c is used as a domain dependent weighting factor between usefulness and age for the relevance. The second constant m serves to synchronize the measurements used for usefulness and time.

$$relevance(a_{i,tc}, u_{i,tc}) = c \cdot forget(m \cdot a_{i,tc}) + (1\text{-}c) \cdot train(u_{i,tc}) \quad (1)$$

The power functions mentioned in the introduction show desirable characteristics for knowledge bases in technical domains. They were therefore a starting point for the "train" and "forget" functions used in our field, which are introduced in sections 4.1 and 4.2.

For each task class the usefulness of information is separately stored so that the relevance for each task class is calculated independently. This method more or less corresponds to the "activation level of memory traces in certain tasks" functions" or in other words a "concentration on a current task". The independent consideration of different tasks prevents conflicting tasks making training of the knowledge base for good solutions impossible. For finding task classes there are two indicators:

The task classes at first result from the combinations of various global optimization objectives, because conflicting objectives would make a knowledge base training for RKF impossible. Therefore if for example the optimization objectives "price" and "performance" are decisive in one domain, the task class can be automatically generated as a result of the combinations of price and performance together with the objectives "high", "low" and "don't care", e.g. "low price / high performance", "don't care the price / high performance" etc.

As a second indicator task classes can be used, which emerge directly from the respective domain. These are for example target groups of customers for configuration systems. The task classes that has been found automatically can therefore be further refined.

As soon as a solution is found, all information which was useful for finding the solution, i.e. all used information, is upvalued. To do this, the solution is assessed and for all used information in the knowledge base its usefulness is increased in relation to this assessment. In this way, information which has lead to good results becomes useful more quickly. The way in which "usefulness" should precisely be defined, depends on the domain. In section 3, we briefly introduce two of our suggested usefulness measurements which can be applied in technical domains.

Information which was seldom or barely useful, becomes irrelevant bit by bit on the basis of its aging. The measurement employed for age also depends on the domain. Section 3 gives some suggestions for the calculation of age.

In order to find good solutions, RKF is used in an optimization loop: As soon as a solution is found it is assessed, the usefulness of the information concerned is increased and the solution process restarts. The random generator that helps in selecting *one of the most relevant* pieces of information, provides the solutions being different. Due to the repeated increase of usefulness that is based on the assessment the optimization loop converges to a global optimum. The loop can be broken as soon as a "good enough" solution has been found or after a given time limit.

Figure 1 shows the knowledge based search algorithm using RKF:

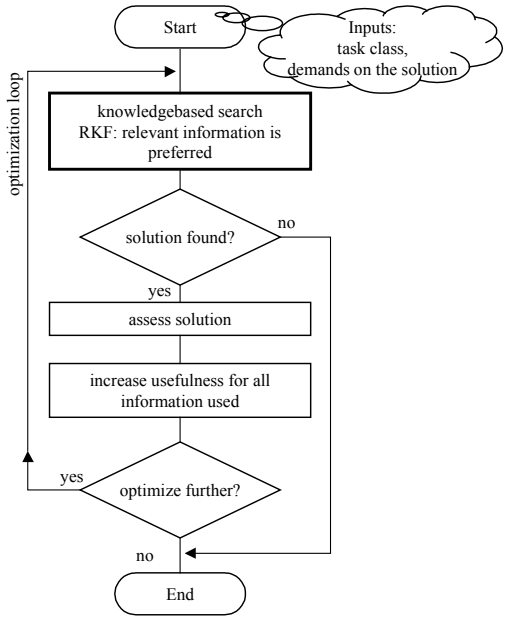

Figure 1. Knowledge based solution process using RKF

3 MEASUREMENTS AND DEFINITIONS FOR THE DETERMINATION OF RELEVANCE

With RKF the relevance of knowledge is calculated as a function of age (time) and usefulness. The measurements for time and usefulness presented in this section serve as a basis for our RKF investigations. In isolated cases they can be further adapted to the area of applicability by which RKF is to be employed

It should be pointed out that relevance of knowledge is only comparable if the same measurements for usefulness and time have been used. Even though several measurements could be applied at the same time. For example a different usefulness measurement can be applied to control knowledge rather than to factual knowledge, in case the distinction between the types of knowledge is made in the knowledge base and the relevance of knowledge of these types is therefore never compared with the other.

Firstly a measurement was sought for the age of information i, which permits comparisons such as "older" and "younger". A relative measurement is therefore sufficient. The age a_i of a piece of information is calculated, as usual, from the difference between the point in time when the information was saved $t_{0,i}$ and the current time t:

$$a_i = t - t_{0,i} \quad (2)$$

Both of the following time measurements have been used in our experiments:

- **RTC** (Real Time Clock): The "real" time of a computer system serves as the current time for the knowledge base.
- **NOR** (Number of Runs): The number of knowledge-based search processes held up to this point in the knowledge base serves as "actual time".

Both measurements have different characteristics. For example using RTC knowledge grows old, even if the knowledge base is not used. If this behavior is not suitable for a domain NOR should be used. An advantage of RTC however is that the semantics of "*outdated knowledge*" is clear i.e. a system's user can, by means of real time, simply decide on the basis of his or her own time feelings.

For the usefulness of information in a knowledge base different definitions can be suitable according to the domain. We have used the following definitions for usefulness measurements:.

- **IPS** (Information was Part of Solution): For each piece of information it is counted how often it was used to help finding solutions. To take the quality of the solutions into consideration the steps are weighted corresponding to the solution's assessment.
- **NAS** (Number of Accesses during Search): This measurement for usefulness also adds up the assessments for the solutions for each piece of information. However in addition to this, the number of times the information in the respective solution process has been accessed is also taken into account.

IPS and NAS both result in a very similar usefulness measurement. The difference is that

- with IPS you are prevented from making a piece of information useful more quickly as a consequence of multiple sub solutions existing in a solution.
- and with NAS precisely this "becoming useful more quickly" will be emphasized.

4 A FUNCTION FOR CALCULATING RELEVANCE

The relevance of information is calculated as a function of its age and usefulness, whereby the age counteracts the usefulness.

In this section the effects on relevance of the "becoming useful" and "aging" processes are separately considered and corresponding functions are stated. This leads us on to a formula for the calculation of relevance which combines both aspects.

4.1 Function for the aging of knowledge

We will now introduce the function forget($a_{i,tc}$) which we propose for technical domains. As a starting point we considered the power law of forgetting (see Introduction). The reason for this is that in technical domains similar characteristics are desirable for the development of relevance dependent on the aging of knowledge as in the human brain:

- New knowledge has maximum relevance and is much more relevant than old knowledge.
- Knowledge loses its relevance faster in the beginning. For example after 10 years one week either way will no longer have a great effect on the relevance of technical knowledge.
- The relevance only approaches zero whereas knowledge is never really irrelevant: the access just lasts longer. Real forgetting, i.e. the irreversible erasure of information is not desirable at first.

To make matters simpler the following assumptions were made compared with the function of the power law of forgetting.

- Reduction in the range of values to [0, 1]: The value for relevance should begin at 100% and approach 0%.
- A reciprocal function shifted to the left by 1 describes the desired effects just as well, such as an exponential function, is however more efficient for computers to calculate.

The first assumption is sensible because many parameters of the Cognitive Psychology are not available in the moment of storing the knowledge. For example in [15] it is described how the start relevance depends on the estimation of the importance of the information. We must assume however that all information in a knowledge base is from the same importance because a computer cannot make "emotional" assessments. The aforementioned "desired" characteristics remain, though.

The following definition describes the forget-function which we applied:

Definition 1:
The process of gradual forgetting information i in a knowledge base in technical domains can be described by the following function:

$$forget(a_{i,tc}): rel_{i,a,tc} = \frac{1}{a_{i,tc} + 1} \tag{3}$$

With

$a_{i,tc} \in [0,\infty[$ age of the information i in the task class tc (4)

$rel_{i,a,tc} \in]0,1]$ the part of relevance that is based on age for the task class tc (5)

Note: As time can be different for different task classes (e.g. when using NOR, number of runs, as the measurement for time) age can be dependent on the task classes. In the above definition this is indicated by the index tc.

4.2 Function for "knowledge training"

The Exponential Learning Function shows characteristics which seem to be adequate in technical domains:

- If information is used often, it seems to be important. The access time should be reduced, that's to say the relevance should be increased.
- The first accesses make information relevant more quickly than later accesses. If for example the same component is used for 10000 configuration processes, ten further accesses no longer have particular importance. Experts have thus learned this component is useful and will try it first.
- Relevance approaches any or the maximum relevance.

We reduce the range of values again to [0, 1] and use the reciprocal function shifted left by one as before in the forget function.

The following definition describes the train function we have used:

Definition 2:
The effect of the "training" of an information i in a knowledge base in technical domains can be described by the following function:

$$train(u_{i,tc}): rel_{i,u,tc} = 1 - \frac{1}{u_{i,tc} + 1} \quad (6)$$

With

$u_{i,tc} \in [0,\infty[$ Usefulness of information i for the task class tc (7)

$rel_{i,u,tc} \in [0,1[$ part of relevance that is based on usefulness for the task class tc (8)

4.3 Relevance of knowledge

In equation (1) the relevance function for an information i results from the addition of the forget and the train function. With equations (3) and (6), the following definition results

Definition 3:
The relevance of information in a knowledge base is described by the following function:

$$relevance(a_i t_c, u_{i,tc}) = c \cdot \frac{1}{m \cdot a_{i,tc} + 1} + (1\text{-}c) \cdot \left(1 - \frac{1}{u_{i,tc} + 1}\right) \in [0, 1] \quad (9)$$

With

$a_{i,tc} \in [0, \infty[$ Age of information i for a given task class tc (10)

$u_{i,tc} \in [0, \infty[$ Usefulness of information i for a given task class tc (11)

$c \in [0, 1]$ Constant for adjusting the weighting of usefulness and age (12)

m Constant for adjusting the measurement of usefulness and the measurement of time used (13)

The values can be interpreted as follows:

Table 1. interpretation of the relevance values.

0% to c	"normal" values for relevance (the range of values of the relevance function is almost always in this range).
c	Start value of relevance for new knowledge and threshold value for old but often useful knowledge.
c to 100%	Above-average relevance, which can only be achieved for a short period of time, if new knowledge is frequently useful right at the beginning.

As already mentioned, the constants c and m serve to weight the forget and train functions relative to each other. This is necessary for the following:

- The time and relevance measurements used must be adjusted to each other
- Depending on the domain age and usefulness can be varyingly important for the solution process

The constant c often can be set to 50% (c= 0.5) to get the same weighting for forget and train.

The following figure shows the relevance function from equation (9) with constants c=0.5 and m=1

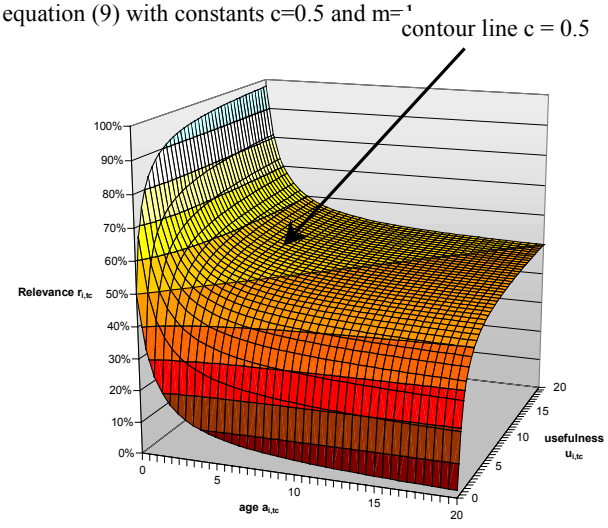

Figure 2. Relevance function with c=0.5 and m=1

As the values for usefulness and age increase to the same extent the relevance stays at c=50% (see Contour line c=0.5 in the figure). If the usefulness grows faster, relevance of over c and up to 100% ("behind" and "left" of the contour line c) can be achieved. Correspondingly the relevance slowly approaches 0% when usefulness grows slower in relation to age.

5 EXAMPLE OF APPLICATION

Before starting a configuration with RKF the objective specifications have to be defined. This happens by selecting a root (possibly also the root of a sub-tree) in the compositional hierarchy and a set of requirements which should be met by the configuration: e.g. functionality, components, parameters or

properties which a configuration should definitely have (e.g. colour "red").

Besides the aforementioned requirements, optimization criteria are typically also part of the objective specification. For example it would be desirable to configure the cheapest, most lightweight or fastest system possible. With RKF the optimization criteria are implicated by the selected task class which is also a component of the objective specification.

Starting from the selected root node of the objective specification it is attempted to recursively define all sub-components from the compositional hierarchy (and in other ways connected components). To this each component is specialized by means of the taxonomic hierarchy until a suitable and constructable part (leaf concept that can be instantiated) has been identified. For every component its parameters also have to be defined. RKF supports a depth first search in the taxonomic hierarchy whereby such leaf concepts are preferred which have a high relevance with respect to the selected task class. The determination of the range of values in the compositional hierarchy i.e. how often a component should be used as a sub-component can be supported by the relevance of specific values with RKF.

After some or even all configuration steps i.e. after every selection and setting of parameters of a (sub-) component, the consistency of the (sub) system must be guaranteed. If a conflict has occurred, it must be resolved e.g. by rejecting a selected component or a set parameter (backtracking). The actual configuration methods differ mostly in how they select components and resolve conflicts (sequence of selecting components, calculation algorithms and heuristics).

The provisional result is a parts list with a structure corresponding to the compositional hierarchy which describes all components together with their parameters, that belong to the configuration (solution). This solution is assessed.

Now the learning phase follows: The usefulness of all included specializations in the taxonomic hierarchy is increased corresponding to the solution's assessment. Equally the usefulness of specific values for the parameters of the components and the compositional relations are increased.

In the case that the configuration found is still not good enough relating to the above assessment, the configuration can be repeated in an optimization loop.

Our first configuration attempt with RKF has been carried out in a relatively simple domain: A PC needs to be configured using RKF from its individual components such as the drives, mainboard, memory etc. To this a depth first search and chronological backtracking has been implemented which is controlled by relevance. Figure 3 shows the compositional hierarchy of this domain.

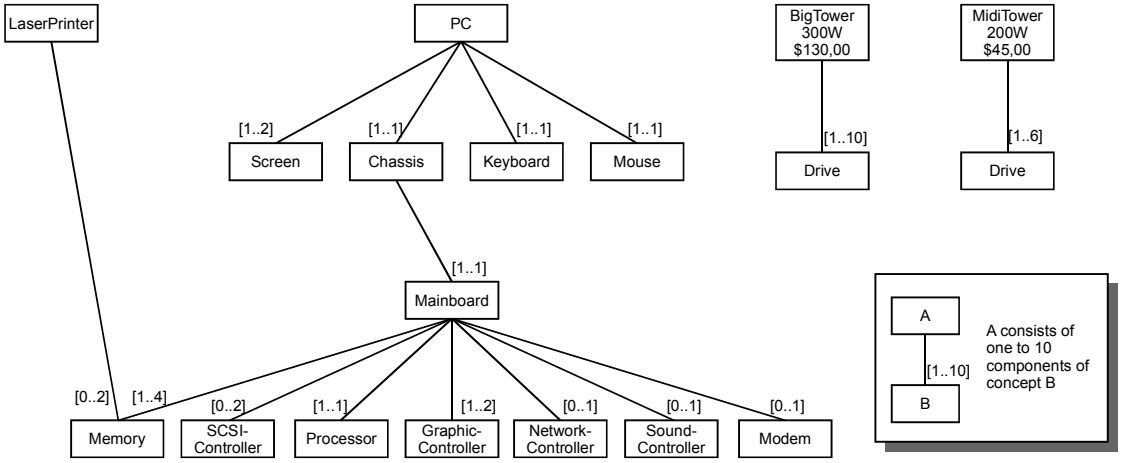

Figure 3: compositional hierarchy of the domain in our example

As a task specification one of these concepts are to be chosen. As a rule this is the concept "PC". Examples of task classes are "Home-PC", CAD-PC" and "Server-PC".

Figure 4 shows part of the taxonomy of the example domain. The above concepts are ancestor nodes of those concepts connected with lines below. For better clarity the alternatives of leaf concepts have been represented as a list one below the other (without further lines).

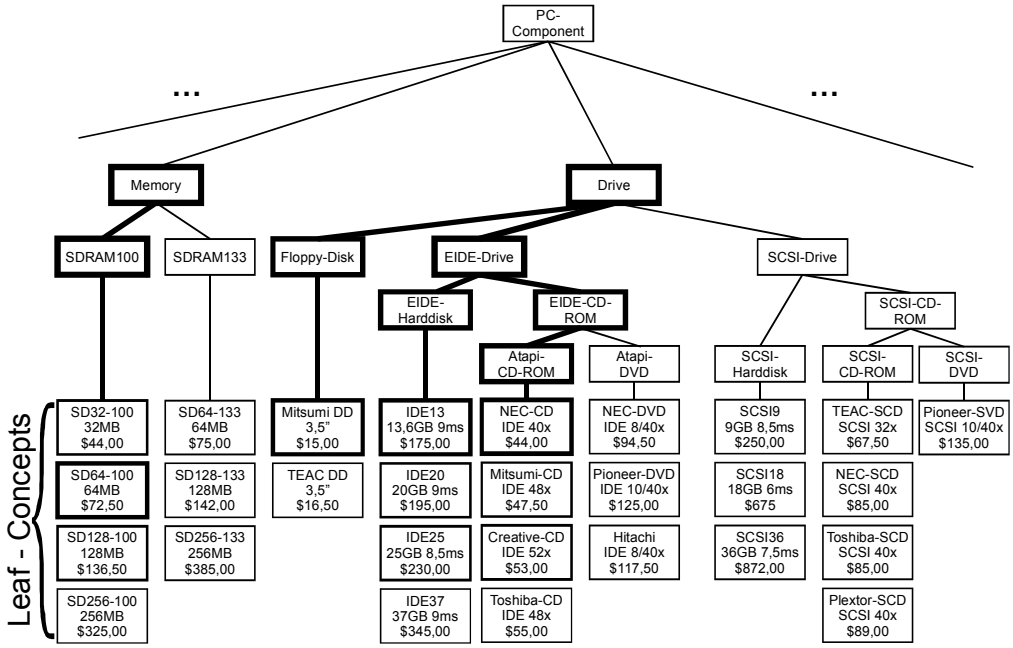

Figure 4. part of the taxonomy of the PC-domain. The thickness of the lines indicate the relevance for the task class "Home-PC"

The thickness of the lines indicate the relevance for the task class "Home PC" i.e. this form of PC most often contains a SD64-100 memory module, a Mitsumi DD-Disc drive, an IDE13- hard drive and a NEC-CD drive. The figure also shows how RKF supports the depth first search in this hierarchy: Paths of high relevance develop because whenever a daughter concept was useful, the respective father concept was also useful. The concept "PC-Component" has very minor relevance because it is not found in the compositional hierarchy. Instead of this the search always begins with a more specific concept (in the "drive" and "memory" examples). In other task classes of the same domain, the relevance can be completely different. For example the relevance of the SCSI hard discs for the "Server PC" task class is much higher than that of the EIDE hard disks.

The flexibility of RKF turns up as soon as a new component appears. Because of its low age it has a high relevance at the beginning despite its small usefulness. In the following example two new hard disk models (IDE22 and IDE27) are added to the taxonomy of the knowledge base.

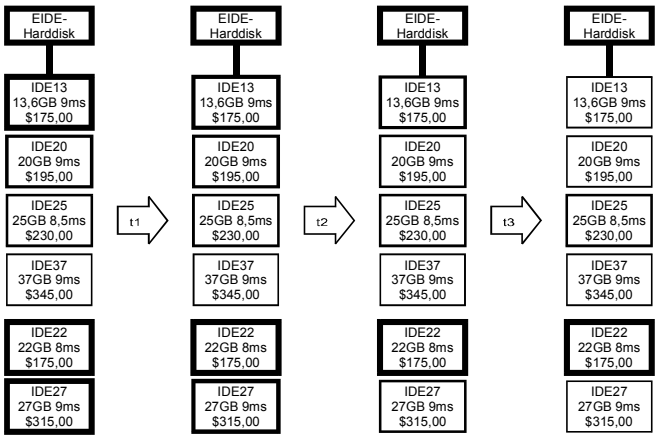

Figure 5. alteration of relevance over time for the task class "Home PC" after adding two new models of hard disks

The thickness of the lines in Figure 5 again represents the relevance of the individual concepts. The two new hard disks have a high relevance at the start because of their low age. Over the time period t1 to t3, the hard disk IDE 22 proves its worth for the "Home PC" task class and bit by bit replaces the hard disk IDE13 which until now was the most relevant, because of its aging without new usefulness. The second new hard disk does not prove its worth and loses its relevance quickly because of aging.

Incidentally the relevance of the father concept "EIDE hard disk" does not change if new concepts are added. That means that for the "Home PC" task class no "relevance trail" will develop for the SCSI hard disks, only due to the fact that a new disk has been added there. RKF reacts conservatively here: It is very unlikely that

RKF would find the new SCSI-hard disk without other heuristics (e.g. user-interaction). On the other hand RKF is flexible enough to adapt to market trends: If it suddenly became "modern" to have SCSI drives within the task class "Home PC", the relevance of these concepts would increase, because of the new demand and finally because of good overall rating. The relevance of the EIDE drives would become less because of aging.

6 SUMMARY AND OUTLOOK

The development of Relevant Knowledge First (RKF) was influenced by the effects of cognitive psychology. First of all age and usefulness were defined for use in RKF, and a function has been specified, which calculates the relevance of information from both these values. The specific characteristic of this method is that knowledge can not only be learnt, but also be "forgotten" without deleting information. This is always an advantage, if the information in a knowledge base is subjected to change. Knowledge bases in technical domains are usually dynamic due to innovations. As an example, the configuration in a PC-domain was presented using RKF. In this example first positive results of this method could ascertained.

Despite the positive findings we have already made using RKF the following questions need to be discussed:

One significant problem is the initialization of the usefulness for the different task classes: Can the initialization be performed automatically? I would say it depends on the way the assessment of the solutions works: If an assessment could be found, that can be calculated without any user interaction this could be achieved. In this case we seem to find a machine that can tell us the future. Because no such oracle has been found yet it might help to have a look at past configuration results. For example PCs sold before the configuration system for PCs was installed.

On the other hand this reminds us of neural networks during their training phase. One could say that the learning part of RKF has some similarity, whereas RKF additionally takes aging particularly into account.

Another issue to discuss, is the definition of the relevance function given in this article. Of course this is only one possibility and yet we have many other ideas which differ slightly. For example for some domains it seems sensible to stress *when* information was successfully used *recently*.

I'm looking forward to the discussion on these issues during the workshop.

REFERENCES

[1] John R. Anderson: *Kognitive Psychologie.* Deutsche Übersetzung von Joachim Grabowski und Ralf Graf. 2. Auflage. Spektrum Akademischer Verlag Gmbh Heidelberg, Berlin, Oxford, 1996, ISBN 3-86025-354-9 Pb. ISBN 3-8274-0085-6 brosch., chapter 6 und 7

[2] Roman Cunis, Andreas Günter, Helmut Strecker*: Das Plakon-Buch. Ein Expertensystemkern für Planungs- und Konfigurierungsaufgaben in technischen Domänen*, Springer-Verlag, Berlin Heidelberg, 1991, ISBN 0-387-53683-3

[4] Marco Dorigo, Vittorio Maniezzo and Alberto Colorini: *The Ant System: Optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man and Cybernetics-Part B, Vol. 26, No.1, 1996, pp. 1-13

[5] Andreas Günter (ed.*)*: *Wissensbasiertes Konfigurieren. Ergebnisse aus dem Projekt PROKON.* Infix-Verlag, Sankt Augustin, 1995, ISBN 3-929037-96-3

[6] Andreas Günter, Christian Kühn*: Knowledge-Based Configuration – Survey and Future Directions – in Knowledge based systems: survey and future directions*; proceedings / XPS-99, Würzburg, Springer Verlag Berlin, Heidelberg 1999, ISBN 3-540-65658-8, pp. 47-66

[7] Andreas Günter, Ingo Kreuz, Christian Kühn*: Kommerzielle Software-Werkzeuge für die Konfigurierung von technischen Systemen* in *KI – Künstliche Intelligenz* Heft 3/1999, arenDTaP Desktop Publishing Agentur, Verlags- und Vertriebs GmbH, Bremen, 1999, ISSN 0933-1875, pp. 61-65

[8] Albert Haag: *Sales Configuration in Business Processes* in *IEEE Intelligent Systems* Jul/Aug 98, 1998, pp. 78-85

[9] Heinrich, Jüngst: *The Resource-Based Paradigm: Configuring Technical Systems from Modular Components* in AAAI-96 Fall Sympos. Series: Configuration. MIT, Cambridge, MA, November 9-11, 1996, pp. 19-27

[10] Ingo Kruez, Thomas Forchert, Dieter Roller: *ICON. Intelligent Configuring System* in Dieter Roller (ed.) *Proceedings of the 31th ISATA*, Volume "Automotive Electronics and New Products", Düsseldorf Trade Fair, Croydon, England 1998, ISBN 0 9532576 5 7, pp. 219-226

[11] Ingo Kreuz, Ulrike Bremer: *Exact Configuration Onboard. Onboard Documentation of Electrical and Electronical Systems consisting of ECUs, Data Buses and Software*, ERA conference 1999, Coventry, UK, 1999, ISBN 0-700806-95-4 pp. 5.2.1-5.2.8

[12] Ingo Kreuz, Dieter Roller: *Knowledge Growing Old in Reconfiguration Context*, in Boi Faltings, Eugen C. Freuder, Gerhard Friedrich, Alexander Felfernig *Configuration, Papers from the AAAI Workshop*, Technical Report WS-99-05, Orlando, 1999, ISBN 1-57735-089-8, pp. 54-59

[13] J. McDermott: *R1: A Rule-based Configurer of Computer Systems* in Artificial Intelligence 19(1), 1982, pp. 39-88

[14] Elaine Rich, Kevin Knight: *Artificial Intellligence, Second Edition.* McGraw-Hill Book Co, Singapore, 1991, ISBN 0-07-100894-2.

[15] David Waltz: *The Importance of Importance* in *AImagazine*, Volume 20, No. 3, Fall 1999, American Association for Artificial Intelligence (AAAI), Menlo Park, CA, USA, 1999, ISSN 0738-4602, pp. 18-35