An integrated Knowledge Engineering Workbench
ACKnowledge Project
Fano Ramparany†, Manfred Aben‡, Luis Solorzano*
†Cap Gemini Innovation,‡University of Amsterdam,∗Telefonica
Tel. +33 76 76 47 47 - E-mail fano@capsogeti.fr

Abstract
This paper presents the Knowledge Engineering Workbench (KEW) developed within the ACKnowledge project. KEW supports the knowledge engineer in all phases of knowledge base systems development, from knowledge elicitation to knowledge base validation. Compared to other computerized knowledge engineering environments, KEW integrates a full range of complementary knowledge acquisition tools. Moreover, KEW architecture allows those tools to be combined in a synergetic way and to support the knowledge engineering process at a fine grain level. An example of using KEW to develop a telecommunication network diagnostic system is also provided to illustrate its functionalities.

1 Introduction
Knowledge acquisition (KA) remains the major bottleneck of the development of knowledge based systems (KBS). The KA process covers extraction, formalization and evaluation of domain expertise that is to be stored in the KB.

Figure 1 shows that the KA process is a critical stage, in the sense that, as the initial step of the knowledge engineering process, it greatly influences the design and implementation of the target KB. For example if an important domain concept has been ignored during the knowledge acquisition stage, it won’t be encoded in the final KB and the performance of the system might be poor while solving problems for which the solution requires the use of this concept.

Another problem is that KA is labour intensive, requiring the knowledge engineer to analyze huge amounts of raw data such as interviews, protocol reports, case description documents and handbooks. This renders the work prone to errors and affects the quality of the final knowledge base.

Thus, much work in the knowledge engineering community is devoted to the development of computer environments to provide assistance to the knowledge engineer during the KA process.

1.1 Current trends in Knowledge Engineer: Environment Development
Current research may be classified into four major categories according to the type of support that the environment offers:

- Methodology support tools (SHELLEY [1], MACHAO [7]) that control the knowledge engineering process and ensure that it progresses within a methodology that covers the whole KBS development life cycle.
- Activity support tools (ALTO [3], SEEK [4]) that partially or fully automate specific tasks faced by knowledge engineer such as concept hierarchy construction of refinement of an existing rule base. Machine learning techniques and tools, belong to this category.
- Expert system prototyping tools or expert system shells, that allow the knowledge engineer to implement knowledge using ready to use knowledge representation languages. This type of tool now reaching a certain degree of maturity, as assessed by the availability of numerous commercial products such as (KEE, ART, CLIPS, OPSS) [5].
- Process support tools (CONSULTANT, AQUINAS) that cover more than one activity, and provide support for coordinating those activities. CONSULTANT toolbox is developed by the MLT Esprit project [6]. However, CONSULTANT includes only machine learning tools. AQUINAS [7] is another example of this type of tool, it integrates repertory grids for the elicitation of domain concepts with analysis tools for the evaluation of repertory grids.

Figure 1. Knowledge engineering process (as modeled by the KADS approach [Broekhuis])
1.2 Our approach

It has been claimed that, for the near future, integrating diverse knowledge acquisition techniques would provide the most promising leverage in the development of knowledge based systems.

The ACKnowledge project aims to provide a conceptual framework for integrating various complementary knowledge acquisition (KA) techniques, including automated machine learning techniques. Exploiting synergy among various techniques allows the KA process to be more efficient both in terms of time and in terms of quality.

The ACKnowledge theoretical approach has been validated through the construction of a Knowledge Engineering Workbench (KEW) and the assessment of KEW in the context of a number of complex applications.

In this paper we introduce the ACKnowledge conceptual framework and present the current version of KEW as an implementation of this framework.

2 Analysis of the KA process

Viewing knowledge acquisition as a modeling process is now widely accepted (see [8]). By this, we mean that knowledge acquisition is a creative process where the knowledge engineer builds a series of models. Each of these models aims at encoding expertise, and should support the reasoning steps during domain problem solving, as well.

From an in-depth analysis it emerged that KA is a step-wise refinement process, that is both incremental and cyclical by nature:

- KA is incremental because each of the expertise models are gradually refined until they are detailed enough to express all the knowledge required to build the target KBS. The status of these models improves at each step with regards to completeness (more and more problems can be solved), consistency (conflicting knowledge should be eliminated), and correctness (solutions derived by the models are correct). KA is also incremental in the sense that the knowledge engineer starts building informal models (such as diagrams without formal semantics) and progresses toward operational ones (such as the target knowledge base to be built).
- KA is cyclical because it involves five types of operations that appear to apply in a regular sequential order:
  a. Evaluation of the knowledge acquired so far;
  b. KA planning of the next steps, on the basis of the evaluation results and of the user goals;
  c. Selection of a technique to repair failures/incompleteness detected during evaluation;
  d. Application of the technique on the data on which a failure/incompleteness was detected;
  e. Assimilation of the knowledge acquired.

This sequence loops by performing an evaluation of the knowledge resulting from the above steps.

Figure 2 shows the cyclical nature of the KA process. Incremental aspect is implied by the assimilation and evaluation steps that result in a richer KB.

![Knowledge acquisition cycle](image)

For a deeper analysis of the KA cycle the reader may refer to [9].

This model of knowledge acquisition has set the basis of the knowledge engineering workbench that we describe in the following sections.

3 KEW Prototype

In order to validate this conceptual framework and provide computerized support to the KA process, the ACKnowledge project has developed a KEW that automates (partially or fully) the preceding operations, in a comprehensive way.

KEW provides facilities for knowledge representation (KR), reasoning, knowledge acquisition, and additional features for guiding the use of KEW and better supporting the knowledge engineering process. We first browse through KEW functionalities and explain how the architecture of KEW integrates those functionalities.

3.1 KEW Functionalities

Functionality of the tools that are incorporated in KEW can roughly be divided into six categories:

- Knowledge representation and reasoning
- Knowledge elicitation
- Machine learning
- Knowledge Evaluation and Integration
- Knowledge Acquisition Support
- Workbench Support
Let us elaborate each category.

### 3.1.1 Knowledge representation (KR) and reasoning

The KR features supported are: frame based knowledge representation, first order logic representation and transformation of frame hierarchies into logical assertions and vice-versa. KEW also provides partitioning mechanisms, that allow modularity of the target knowledge base. Editing facilities allows the user to directly enter or modify pieces of knowledge contained in any partition. A browser provides a graphical interface to the knowledge partitions.

From the reasoning point of view KEW supports inheritance of properties, active facets and theorem proving, and reason maintenance.

### 3.1.2 Knowledge Elicitation (KE)

This type of tool supports the knowledge engineer in eliciting knowledge from the expert and structuring this knowledge. KE tools available in KEW include:

- Protocol Editor, which allows the knowledge engineer to record interviews, “think aloud” protocols, and written material and to identify important domain concepts, terminology and ways of reasoning in the domain.
- Lexicon editor, which allows the knowledge engineer to manage the list of domain-specific terms and to specify translations and synonyms.
- Card Sort Tool, to elicit important attributes of concepts.
- Repertory Grid Tool, to assess the specificity of the concepts, and acquire extra knowledge to distinguish between concepts.
- Laddering Tool, to elicit concepts in a structured way, thus also eliciting the structure in which the concepts are related.
- Conceptual Model Editor, to specify and refine the reasoning that goes on in the domain in a structured, implementation independent way (a la KADS)
- Control Knowledge Editor, to elicit the control over the primitive inference steps.

### 3.1.3 Machine learning (ML)

ML tools do automatic acquisition of knowledge. The current version of KEW incorporate a similarity based learning tool implementing the AQ algorithm. Given a set of case-descriptions from the domain, this tool derives classification rules for the concepts implied by the cases. Other typical examples of such tool that might be integrated in future versions of KEW are:

- Explanation based learning tool. Given knowledge already acquired, this tool generalizes the knowledge to improve its problem solving capacity in terms of scope or efficiency.
- Case based learning tool. Given a set of problem solving cases, this tool maintains a set of generalized case-descriptions, adjusting it with each observed case, to improve problem solving behaviour of the cases.

### 3.1.4 Knowledge Evaluation and Integration

These tools assess the quality of the knowledge acquired, in terms of problem solving capacity (efficiency, completeness, correctness) or other aspects, such as readability. They also do knowledge integration which encompasses the removal of inconsistencies between acquired knowledge, and the transformation of acquired knowledge to other representations. This transformation is usually done by the tools themselves. So, if one tool operates on frames, it will have a functionality to transform these frames to first order logic formulae. Transformation requires elicitation, and involves the possible loss of information. Therefore transformation is usually irreversible.

Tools included in KEW are:

- Logic debugger (HCD). As one of the core representations in KEW is first order predicate logic, this tool is included to observe the dynamic behaviour of the logical theories. It monitors the proof procedure, and allows the user to adapt the observed behaviour towards the intended behaviour. The user can inspect and alter a generated proof tree in various ways, and determine how the knowledge should be altered to achieve the desired improvement of the problem solving capacity of the knowledge.
- Frame validation tool (KVAT). The other core representation in KEW is a frame-based language. This tool uses test cases to assess the problem solving quality of the frames.
- Frame integration tool (KIT). This tool detects inconsistent or redundant information between sets of frames, and merges hierarchies of frames. The knowledge represented in first order logic is integrated by means of a reasoning maintenance system incorporated in KEW.

### 3.1.5 Knowledge Acquisition Support

What distinguishes a workbench from a toolbox, is the fact that the tools are presented to the user in a guided, cooperative way. A workbench should be able to advice the user which tools to use and guide the user through the knowledge acquisition life cycle. Tools in KEW to support knowledge acquisition are:
• Knowledge acquisition goal planner (KA planner). Which goals are there to be achieved (most top-level goal would be: "Acquire enough information to build a successful knowledge based system") and how can they be achieved. This planning is a typically reactive planning process. One might for instance start with initial scoping, and the creation of a conceptual model of the tasks to be achieved by the artifact. The kind of conceptual model will typically influence the way to proceed, e.g. acquire concept hierarchies or cases. The planner decomposes the goals into operational chunks, i.e. goals that can be achieved by applying some technique. This technique can be applied by using some configuration of tools, and the system should give advice which configuration is most suitable.

• Knowledge acquisition monitor. This tool supports the execution of KA plans produced by the KA planner. Although global control decisions (such as whether he will finally use the technique proposed by the planner) are left to the user, coordination among the operations scheduled by the planner might be handily automated. This coordination is achieved by the KA monitor.

For example, transfer of data produced by one KA-tool to the next tool which might use this data as input is carried out automatically by the KA monitor.

3.1.6 Workbench Support

The tools in this category are those that facilitate communication between tools, and inspect the knowledge acquired. KEW provides the following tools:

• Common information repository browser. The knowledge in KEW is distributed over partitions. This tool allows the user to inspect all partitions and navigate through all the knowledge acquired. The user can look in detail at the knowledge, and the relations between knowledge objects.

• Selection server. The workbench offers rich facilities to select objects and import them into other tools, or perform operations on them.

• Knowledge Transformation facilities allow the translation of tool specific knowledge representation into a common knowledge representation, so that knowledge acquired by one tool may be reused or refined using another tool.

Further details on the tools may be found in [10]

In the next section we explain how those functionalities are organized and cooperate toward the global goal of performing efficient knowledge acquisition.

3.2 KEW architecture

KEW is organised as shown in figure 3. The functionalities introduced in the preceding section interface to the data model and the KEW top level driver.

The data-model provides the generic class definitions shared by the tools. Current classes include partitions for storing pieces of Knowledge acquired so far (KB-partitions or Intermediate Information Structures (IIS)) and generic data-types used by different tools such as dictionary terms. Partitions and IISs may be inspected via a dedicated CIR browser.

The infrastructure of implementation that operationalizes the data model allows partitions and their contents to be persistent. Thus knowledge acquired during an earlier session with KEW can be stored and retrieved later when needed.

The KA monitor controls the execution of the KA operations involved in the current KA plan. The KEW top level driver displays the top level User Interface (UI) layout for KEW and allows the user to activate KA-tools, support tools, and CKB tools in a uniform way, preserving a consistent “look and feel” among the different tools.

4 Example of the use of KEW

The KEW prototype has already been used in the development of EXPERPAC, an Expert System for diagnosis
of the Spanish public data communications network. We have already discussed before the KEW functional architecture and here we will illustrate how KEW could help in supporting the KA process, with examples taken from a real life knowledge engineering context (EXPERPAC).

4.1 KA Planning and Tool Selection

In an initial situation with an empty knowledge base, the first step in the use of KEW is to define a KA process plan. A KA plan consists in a description of KA tasks/goals and can be generated by KEW planner facility. The suggested plan is obtained by applying a set of rules that describe a KA theory and a set of facts that describe the KA context. However, the current prototype includes only a version with limited functionality.

The KA plan description also serves as a reference to monitor the current state of the KA process. The KA planner and monitor allow the generation of a KA plan, selection of tools and monitoring of KA activities.

The KA plan is an and/or tree whose nodes represent a KA goal or task to be performed. In this case, the goal tree is generated automatically by KEW function "Propose KA Plan" on request from the user. When one of these goals is selected, the KA cycle agenda is updated. We can also obtain advice and guidance for selection of tools. For example, after selecting the KA goal "Acquire Symptom Space" when we invoke the "Generate KA Cycle" function KEW shows an instance of the generic KA cycle described earlier. That is, a proposed sequence of tools to be applied. In the figure 4 we can see an example of the agenda, in the context of the EXPERPAC development. Three tool sessions are planned. Namely, a laddered grid named alto, the knowledge integration tool p-kit and the knowledge validation tool kvat. As shown in the figure, the user is ready to "activate" alto whose corresponding KA operation has been selected. However, this is only a recommendation and therefore the knowledge engineer always decides what to do next. (S)He may activate any KA operation defined in the agenda or (s)he can choose any other tool directly from the KEW top level driver. When the tool application has finished, the description of the status in the agenda is updated to reflect either that the operation has been done or the tool invocation failed.

4.2 Knowledge Elicitation

Acquiring new knowledge is supported in KEW by several tools that implement elicitation techniques like laddering grid, card sort, repertory grid and machine learning techniques like AQ. Since the tools are integrated in a workbench, the cooperation between different tools is possible.

Following the previous example, in which KEW proposed a laddering tool for acquiring a classification of symptoms, we decide not to follow this recommendation and invoke the card sort tool. This is because the expert does not have a clear criteria for classifying the symptoms. We invoke the tool and the expert sorts the cards according to the dimension "origin" into three groups: operator, system and subscriber. The significance of this classification is related to the agent who provided the symptom, which is an important factor in the EXPERPAC domain.

The resulting classification is transformed into SFL frames, and we then activate the laddering tool (in the KA cycle agenda) to complete the hierarchy of symptoms. The transformation is needed here because the laddering tool requires SFL frames as input whereas the card sort tool output does not use this representation language. This episode illustrates the synergy between knowledge elicitation tools. However, the cooperation between machine learning and knowledge elicitation tools is also possible.

In another step of the KA process we select the goal "acquire hypothesis generation knowledge". This corresponds to heuristic rules that infer hypothesis from symptoms. We first run a repertory grid session to elicit the relevance of each symptom for a set of initial faulty states (hypothesis). The rules resulting from the entailment analysis in the repertory grid are used as background knowledge to run a session of machine learning with AQ on a set of cases. A refined set of rules is obtained that suggest hypothesis from a set of symptoms. This set of rules is also used by AQ as preference criteria for gener-
ated hypotheses. This may seem an unusual application of a repertory grid tool, however, we found it useful and it shows a possible way of combining elicitation and machine learning tools.

Figure 6 Elicitation by combining repertory grid tool and AQ

4.3 Knowledge Integration and Refinement

Knowledge is acquired from different knowledge elicitation or machine learning tools. However, the Core Knowledge Base in KEW use NTP and SFL as knowledge representation languages. Therefore, we need knowledge transformation tools in order to translate the knowledge acquired in tool-specific representation into a common language. The diverse sources of that knowledge makes also essential the existence of knowledge integration, validation and debugging tools, like KIT, KVAT, HCD.

In a previous episode we obtained a hierarchy of symptoms using the card sort and laddering tool. This new hierarchy is different from the knowledge in the current CKB. Figure 7 displays the application of KIT to integrate both pieces of knowledge into a unique partition of the CKB. In this example, the hierarchies of symptoms differ on the name of a concept and on their attributes. The tool analyzes both hierarchies and detects that "symptoms" is the plural form of "symptom" and then the user is asked whether they are the same concept or not. The tool also compares attributes defined for both hierarchies and merges them in such a way that the common attributes are preserved and different ones are added.

After this integration episode we obtain a new integrated knowledge base and we need to validate the knowledge base. In the next episode, the knowledge engineer wants to check that all the tests associated with the defined faulty states have been defined and are the ones the expert should perform. The KVAT tool is applied at this stage. We define validation cases from the problem-solving cases provided by the expert. The tool tries to match the contents of the knowledge base with the testing examples. The tool finds some inconsistencies between the tests associated to some faulty states and the expert behaviour (see fig. 8).

Finally, the Logic Debugger is used to refine and edit a theory for diagnosing by causal reasoning. We represent as NTP rules all the knowledge needed to make causal reasoning and we debug it with that tool. The HCD shows a proof tree (see fig. 9) and we compare the behaviour of the knowledge base with the expert behaviour. The modifications of the proof tree by the user imply changes in the NTP partitions under the supervision of the tool.

5 Discussion and Future Plans

KEW is an integrated knowledge engineering workbench that supports the Knowledge Acquisition process in the development of Knowledge-based systems. Knowledge can be acquired from different sources using a variety of knowledge acquisition tools: knowledge elicitation, machine learning and editing/browsing tools. KEW includes a number of such KA tools and provides integration facilities between them. The "value-added" of KEW is the possibility of cooperation between these tools. Putting
together existing tools is not enough for making a workbench. Knowledge transformation and integration is a crucial facility for KEW because allows to use the best suitable tool to acquire each type of knowledge without worrying about the output representation language used by the tool/technique.

KEW demonstrates that it is possible to use existing tools in order to create an integrated workbench. We have constructed a system using existing tools and techniques. They were reimplemented using a new common infrastructure that ensures homogeneity in certain aspects: user interface, sharing of resources, knowledge-bases management, communication between tools, ...

From the knowledge engineer's point of view, KEW may help to obtain the knowledge using the best method/technique available. And if the tools applied to acquire knowledge use different representation languages you can transform their output into a common knowledge representation language and integrate it in a core knowledge base. Therefore, KEW does not restrict the use of tools to those which share a common knowledge representation language.

The current prototype of KEW has some limitations. The tools in KEW are now loosely integrated. It is possible to transform from a tool-specific into a common knowledge representation language. However, there is no direct transformation between tools. The communication between two tools is not possible if they do not share the same representation language. Those tools which use different languages have to relay on the transformation mechanisms to cooperate.

Finer-grained integration of tools will be one of the improvements for the next prototype of KEW.

In toolboxes the distinction between tools is usually very clear, they easily fall into one of the categories described above. In a workbench, however it is not always possible to distinguish between categories. It is moreover not always very desirable to do so. In [11] it is argued that it can be quite limiting to be talking in terms of tools (implementations of techniques) instead of functionalities (to be achieved by the techniques, or parts thereof). The similarity based learning tool incorporated in KEW, for example, elicits the domain specific elements of the language which describes the example cases the tool is to learn rules from. As it learns, it constantly evaluates its hypotheses to see whether each hypothesis covers all positive examples and no negative examples. Therefore, it seems rather contrived to fit the tool into one specific category. However, the tool seems to entail functionalities which can be placed into these categories. Having dropped the notion of tools, and replaced it by the notion of functionalities, fine-grained integration of these techniques becomes possible. Depending on the domain, the elicitation of the example language in the tool just described, could be performed by various ways (e.g. card sort, ladderising or repertory grid). The evaluation could be performed using the logic debugger. The other way round, the SBL tool could be used to implement changes to the knowledge bases proposed by the debugger. This fine-grained integration of functionalities does have its consequences for the integration of knowledge representations. We refer to [11] for an extensive description of these consequences.

Other interesting improvement for the next version will be the advice and guidance component. It will be based on models of expertise, like the conceptual models of KADS methodology [12]. Model-driven knowledge acquisition will benefit the user in that the knowledge engineer will be assisted by KEW regarding what knowledge should be acquired and which tools are best suited for acquiring each piece of knowledge.

6 Acknowledgement

We wish to thank Geir Aakvik, Anjo Anjewierden, Sergio Calabretta, Pete Chalmers, Ray McDowell, Jorun Eggen, Gertjan Van Heijst, Nigel Major, Ole Jakob Mengshoel, Fabrice Mrugala, Han Reichgelt, Peter Terpstra, Mette Vestli, Sandra Whitehouse, for their work on implementing KEW-1, Pedro Luis Munoz and Emilio Rodriguez for their participation in evaluating KEW-1 on EXPER-PAC application, Jean-Charles Marty, Cléo Julien and Georges Jetatis for their valuable comments.

This work was partially supported by Commission of the European Communities as ESPRIT project 2576. The partners in this project are Cap Gemini Innovation (F), General Electric Company (UK), Telefonica (E), Computas Expert Systems (N), University of Amsterdam (NL), SINTEF (N) and University of Nottingham (UK).
Bibliography