Knowledge Acquisition (1994) **6**, 115–136

An integration of knowledge acquisition techniques and EBL for real-world production planning

THOMAS REINARTZ AND FRANZ SCHMALHOFER†

German Research Center for Artificial Intelligence, Erwin-Schrödinger-Strasse, 67663 Kaiserslautern, Germany

The paper presents an approach to the integration of knowledge acquisition (KA) techniques and explanation-based learning (EBL). Knowledge acquisition techniques are used to delineate a problem class hierarchy for different manufacturing tasks in mechanical engineering. This hierarchy is stepwise formalized into a terminological representation language. The terminological descriptions are then combined with cases of specific manufacturing tasks and their solutions (in the form of production plans). Explanation-based learning is applied to the cases and skeletal plans are automatically constructed for the terminal classes of the problem class hierarchy. Such skeletal plans consist of a dependency structure with a sequence of operators, that can be instantiated to specific plans for all other problems of the class. An evaluation of the proposed KA/EBL integration demonstrates its strengths as well as certain limitations of explanation-based generalization.

1. Introduction

Although the fields of machine learning and knowledge acquisition share the common goal of developing operational knowledge-based systems, there are few systems where the specific advantages of automatic machine learning methods (e.g. Mitchell, Keller & Kedar-Cabelli, 1986; DeJong & Mooney, 1986), model-based knowledge acquisition techniques (e.g. Breuker & Wielinga, 1989) and knowledge acquisition from human experts (Boose & Gaines, 1989) have been combined. Apprenticeship learning systems like LEAP (Mitchell, Mahadevan & Steinberg, 1985), DISCIPLE (Kodratoff & Tecuci, 1987; Tecuci & Kodratoff, 1990) and PROTOS (Porter, Bareiss & Holte, 1990) were already successful in combining analytic and inductive machine learning procedures with knowledge elicitation tools and knowledge editors. Apprenticeship learning systems acquire new knowledge during the course of problem solving. They typically consist of a preliminary knowledge base, a problem solving component and an apprentice learning component. The system takes as its input the normal problem-solving actions of a human expert without any explanations. The solving of a new task is then performed by the problem solving component of the system and the human expert provides advice. Alternatively, the human expert may perform a task and the apprentice system learns through observation.

A Case-Oriented EXpert or COEX architecture (Schmalhofer & Thoben, 1992) was developed, in which model-based knowledge–engineering techniques (Kühn & Schmalhofer, 1992; Aitken, Kühn, Shadbolt & Schmalhofer, 1993), explanation-based learning (Schmalhofer, Bergmann, Kühn & Schmidt, 1991), and knowledge-acquisition from human experts are integrated in a thoughtful way: rather than

† Author to whom correspondence should be addressed.

1042–8143/94/020115 + 22$08.00/0

simply applying some favourite techniques or tools, we thoroughly analysed the complex application domain of production planning in mechanical engineering. From the availability of different information sources and the accessibility of human experts, we then devised an integrated knowledge acquisition method (Schmalhofer, Kühn & Schmidt, 1991).

Although there are many similarities to apprenticeship learning systems, the COEX architecture shows a number of important differences to systems like LEAP, DISCIPLE and PROTOS. Unlike these systems, which learn individual solutions during the normal use of the problem solving component, in COEX reversed engineering is applied to completed success cases from the real world. From these cases solution schemata are obtained, which are then employed for top-down reasoning to solve future problems. Instead of the very interactive apprentice–master relation between apprenticeship learning systems and its users, in COEX the expert is viewed more as an author who instructs the system. The system, one the other hand, plays the role of an editorial assistant. The human expert and the system thus cooperate in authoring a knowledge-based system.

In this paper, we will first describe the application and task domain and show how their characteristics lend themselves to the integration principles of the COEX architecture. Two major components of the COEX architecture, namely the Case-Experience Combination System (CECoS) and the Skeletal Plan GENeration procedure (SP-GEN) are then presented in detail. CECoS is a type of knowledge editor. Model-based knowledge engineering, a clustering algorithm, global expert judgments and natural language expressions are jointly used to outline and stepwise formalize a hierarchy of problem classes into a KL-ONE-like representation (Brachman and Schmolze, 1985). By employing explanation-based learning, SP-GEN then generates skeletal plans with a dependency structure between its individual parts for the defined problem classes. As it turns out, SP-GEN is well suited to constructing such plans for the terminal classes. However, it is not appropriate for the non-terminal problem classes which are described in more abstract terms. This limitation and the integration achieved with CECoS and SP-GEN will also be evaluated and discussed.

2. Complexity of task and domain

The planning process for the manufacturing of rotational parts is a complex task, which plays an important role in industry. Companies like the Feldmühle AG do not only sell manufacturing machines, but also provide prototypical as well as customized production plans for different products (Feldmühle, 1984). Human experts require several hours to produce an initial plan. A total of 2 days is spent before it is successfully tested and a qualitatively good plan is obtained. This complexity arises from the enormous number of technical details which cannot be neglected and a large number of only vaguely known interdependencies among these knowledge items. Only human experts have sufficient experience to provide an appropriate classification of the various problems. We will now exemplify the complexity of this domain for the task of manufacturing rotational parts.

Rotational parts or workpieces are manufactured by putting some more or less cylindrical piece of metal (mold) into a fixture (i.e. chucking) of a manufacturing

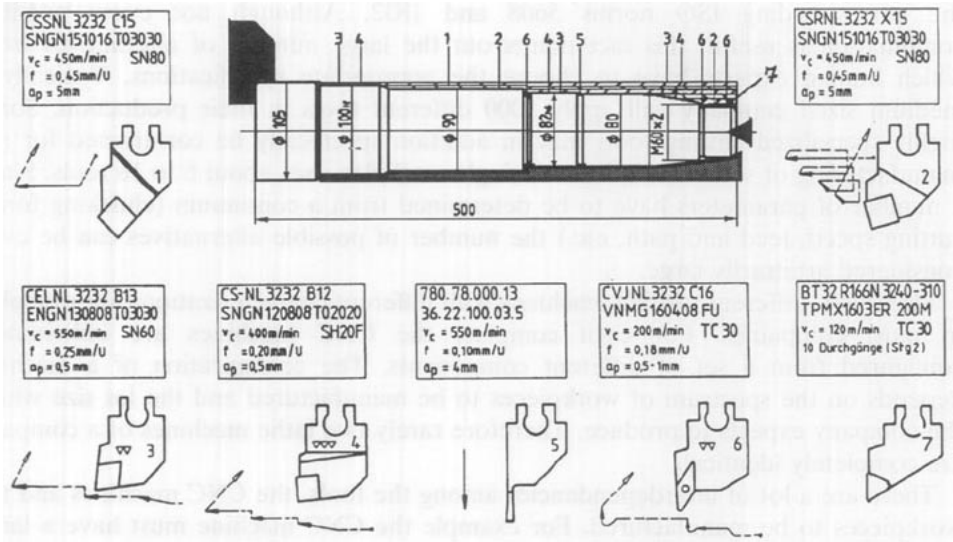

FIGURE 1. Graphical representation of a typical production plan (From "*Examples for application*" p. 22, Plochingen Neckar, Germany: Feldmühle AG. Copyright 1984 by Feldmühle AG. Reprinted by permission.)

machine (CNC machine). The chucking fixture together with the attached mold is then rotated at a relatively high speed, with the longitudinal axis of the cylinder as the rotation center. The rotational axis and all movements of a specific cutting tool (movements which perform a cut as well as movements which only position the tool) lie in the plane. While the chucking fixture and the attached mold are rotated, a cutting tool moves along some contour and removes material according to a given production plan. thereby the desired geometric shape of the workpiece is obtained. As a result of this processing, axle shafts, drive shafts or bevel wheels may be produced.

Figure 1 shows the graphical representation of a typical production plan (Feldmühle, 1984).† In this figure the initial mold (shaded grey area) and the desired goal workpiece are overlayed. The chucking fixture is indicated by black areas on the left and right hand of the workpiece. Each different cutting step to manufacture this workpiece is numbered from 1 to 7 and also presented including its detailed specifications. For example the first rough cut is done by tool "CSSNL 3232 C15 SNGN151016T03030" (cutting material SN80, namely silicium) by a rotation speed of $v_c = 450\,\mathrm{m\,min^{-1}}$, a feed of $f = 0{\cdot}45\,\mathrm{mm\,U^{-1}}$ and a cutting depth of $a_p = 5\,\mathrm{mm}$. In addition a complete production plan would contain more technological data of the workpiece (surface roughness, material, tolerances, etc.) and precise workshop data (CNC machines with their rotation power, number of tools and revolvers, etc.). This plan is relatively simple in comparison to the planning processes by which it was generated. The following facts may explain why these planning processes are so time consuming.

Theoretically $1{\cdot}8 * 10^7$ toolholders and $1{\cdot}5 * 10^8$ tool inserts can be specified within

† Throughout the paper, this case will be denoted by g5w1d3.

the corresponding ISO norms 5608 and 1832. Although not every possible combination is useful, this fact points out the large number of alternatives from which human experts have to choose the appropriate specifications. Typically a medium sized company will apply 5000 different tools in their production. Some highly specialized cutting tools may in addition specifically be constructed for the manufacturing of some workpiece. A single workplan uses about 6 to 16 tools. Since a number of parameters have to be determined from a continuum (chucking force, cutting speed, feed and path, etc.) the number of possible alternatives can be even considered arbitrarily large.

Moreover, different turning machines with different characterizations are available in most companies. For each company the CNC machines are individually configured from a set of different components. The configuration of a machine depends on the spectrum of workpieces to be manufactured and the lot size which the company expects to produce. Therefore rarely two lathe machines of a company are completely identical.

There are a lot of interdependencies among the tools, the CNC machines and the workpieces to be manufactured. For example the CNC machine must have a large enough revolver to keep all the necessary tools, and the machine power has to be sufficient to achieve the required cutting speed specified in the production plan. It is therefore not surprising that much time and effort must be invested by human experts in order to develop a qualitatively and economically good production plan.

3. Integration principle

Using a model-based knowledge-engineering approach (Breuker & Wielinga, 1989), hierarchical skeletal plan refinement (Kühn & Schmalhofer, 1992; Aitken *et al.*, 1993) was developed as a KADS-interpretation model. This model assumes a hierarchy of problem classes, where skeletal plans are associated with each class in the hiearachy. A set of prototypical cases which delineates the desired competence of the future knowledge base and the judgments of a domain expert about their relatedness are used to build a problem class hierarchy by conceptual clustering. Unlike COBWEB (Fisher, 1987) where the clustering is performed on the basis of the descriptions of the specific cases, our clustering relies upon the expert judgement about deep relationships. The surface form of the case description does therefore not bear much consequence. The formation of the class hierarchy is thus independent of any incidental selection of a specific representation language but grounded in an expert's understanding of the domain.

A representation of the individual cases and the problem classes are subsequently produced by a stepwise formalization of the expert's natural language descriptions of cases and problem classes. These descriptions consist of lists of features that are attributed to the different problem classes. Several templates are used to establish an adequate and more fine-grained semantic structure of the various features. The templates were constructed so that each template can be independently translated into the terminological language TAXON (Hanschke, Abecker & Drollinger, 1991). With the inference services of this KL-ONE-like representation language the class hierarchy is verified and new production problems can be classified.

In a similar vein, a class hierarchy is established for various operators which occur

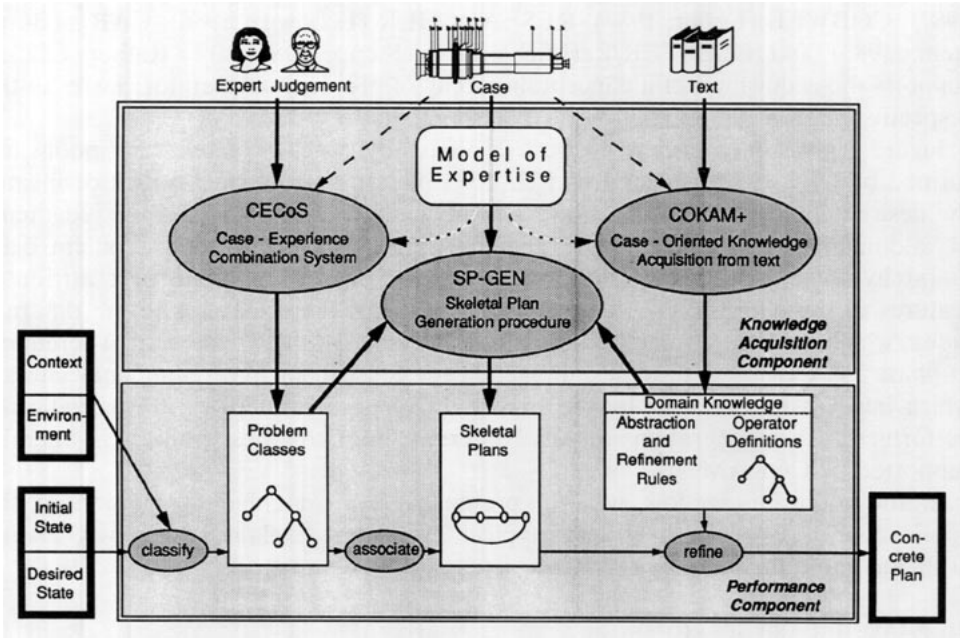

FIGURE 2. Schematic diagram of how the proposed integration of machine learning and knowledge acquisition techniques has been applied in the COEX architecture.

in the different production plans. After the knowledge base has been established so far, explanation-based learning is applied to generate skeletal plans for the various problem classes. Specific cases serve as the examples.

The proposed integration of knowledge acquisition techniques and machine learning is supported by the model of hierarchical skeletal plan refinement, the knowledge acquisition tool CECoS (Reinartz, 1991; Tschaitschian, 1991), and the machine learning tool SP-GEN (Schmalhofer *et al.*, 1991). The integration of these tools for developing a case-oriented knowledge-based system is shown in Figure 2 and further described by Schmalhofer and Thoben (1992). Figure 2 shows the knowledge acquisition and performance components of the COEX architecture: CECoS is guided by the expert and constructs a hierarchy of problem classes from a set of cases. From the same cases, skeletal plans are formed by SP-GEN. Additional background knowledge is acquired from text with the tool COKAM+ (Schmidt, 1992). The performance component shows how a new problem is first classified, and the associated skeletal plan is retrieved. By refining this skeletal plan the solution for the specific production problem is then obtained.

4. Hierarchical clustering and terminological definitions of problem and operator classes

The knowledge acquisition tool CECoS is not an automatic concept formation procedure as the GCC-algorithm (Maarek, 1990), UNIMEM (Lebowitz, 1986;

1987), COBWEB (Fisher, 1987), RESEARCHER (Lebowitz, 1986), CARL (Burnstein, 1983) and CLUSTER/2 (Michalski & Stepp, 1983) are. Rather CECoS supports the acquisition of a concept hierarchy of problem or operator classes in the respective application domain.

Figure 3 gives an overview of the five phases by which problem descriptions are formed by CECoS. In phases I and II, a set of prototypical cases which delineates the desired competence of the future knowledge base is selected and the judgments of a domain expert about their relatedness are used to form a problem class hierarchy by a hierarchical cluster analysis. In phase III, the expert attributes features to the different problem classes and assigns these features to the different views, which were determined by the model of skeletal plan refinement. Thereafter, in phase IV, CECoS determines an agenda for consistency and redundancy checks, which have to be executed by the expert. In phase V a stepwise formalization is performed. The generation of the different operator descriptions is thereby supported by COKAM+.

In the following section, we will present a more detailed description of the application of CECoS to 60 prototypical cases and explain how newly arising problems can afterwards be classified.

4.1. PHASE I: CASE SELECTION

Within the first phase appropriate exemplar cases are selected. This selection should be performed according to the needs of the specific application area (e.g. in a company). Although the selection phase is presented as part of CECoS, this set of cases also serves as the examples used to construct skeletal plans with SP-GEN.

The reported example application is based on 60 production problems characterized in three dimensions: geometry, workpiece material and (CNC) turning machine (which has been used to manufacture the desired goal workpiece). The chosen geometries (g1–g5) contain two drive shafts (g1 and g2), one pinion shaft (g3) and two axle shafts (g4 and g5). An unalloyed steel (w1), a special kind of cast iron (w2), a certain type of aluminium (w3) and an alloyed steel (w4) are the different workpiece materials (w1–w4). Finally three different machines or lathes (d1–d3) had been available: a lathe at the low end of the performance spectrum (5·5 kW) with only one revolver and four tool holders (d1), a lathe (70 kW) with one revolver and six tool holders (d2), and a high performance lathe (90 kW) with two revolvers and 12 tool holders (d3). The production problem shown in Figure 1 is consequently denoted by g5w1d3.

4.2. PHASE II: PAIRED COMPARISON AND HIERARCHICAL CLUSTER ANALYSIS

Each pair of exemplar cases is presented graphically to a domain expert. Thereby the surface description of each object is visually presented, in a way typically used in this application domain. Now, the expert judges the similarity between the presented cases on a discrete scale between 1 (very dissimlar) and 7 (very similar). Thereby the expert guides the classification. Since a complete pairwise presentation yields $n*(n-1)/2$ comparisons for n objects, this complexity was reduced by preclassifying the cases in primary step. CECoS is then applied to the prototypes of the different classes. Thereafter, it is separately applied to all the objects of every class. The resulting tree structures are subsequently joined.

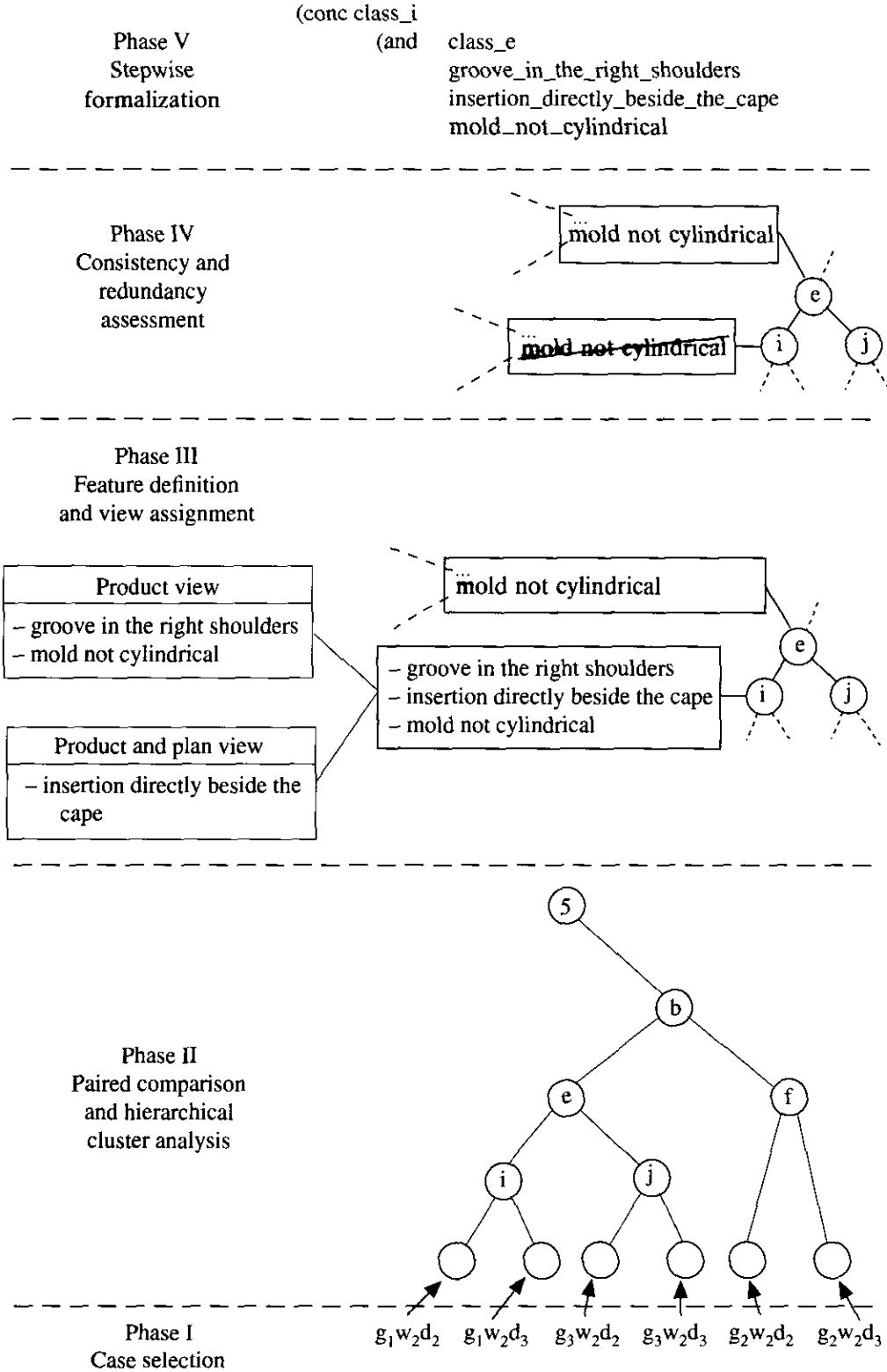

FIGURE 3. The five phases of the CECoS procedure are illustrated by an example application.

In the primary step the 60 example cases have been categorized into 11 different classes. These classes are denoted by the numbers $0, 1, \ldots, 10$. The application of CECoS will be described for the objects of one of the 11 classes. Note that the problem class hierarchy for this class is only a subtree of the whole class structure. The nodes of this subtree are denoted by letters.

A hierarchical cluster analysis is applied to the resulting similarity matrix of the pairwise comparison. Hierarchical cluster analysis is one of several methods for conceptual clustering along with optimization, clumping, incremental clustering, etc. (Backhaus, Erichson, Plinke, Schuchhard-Ficher & Weiber, 1987). CECoS uses the agglomerative clustering method of "Nearest Neighbor", i.e. the most similar objects are clustered subsequently, until a single problem class contains all objects. Each node within the resulting concept hierarchy represents a single problem class. Each terminal node contains at least one specific exemplar case. Within the hierarchy the subsumption principle holds, i.e. each class subsumes its successor classes. The bottom section of Figure 3 depicts a subtree of the problem class hierarchy. Specific cases as g1w2d2, g1w2d3, g3w2d2, etc. are prototypical members of the terminal problem classes.

Following the paradigm of user guidance the expert is now asked to "verify" the problem class hierarchy. The hierarchy may be accepted immediately without any comments, but can also be edited. Therefore errors in similarity judgments may be corrected, and it is ensured that the delineated categories are meaningful to the expert.

4.3. PHASE III: FEATURE DEFINITION AND VIEW ASSIGNMENT

In the third phase, the expert is asked to name the features which define the different problem classes. Each feature is a natural language expression and can be seen as a single knowledge unit representing some term in the domain terminology. The conjunction of the defining features of a class delineates an explanation for the respective problem class. For example, for problem class i the expert named the following features: "groove in the right shoulders", "insertion directly beside the cape", and "mold not cylindrical". In addition, class i inherits the defining features from classes 5, b and e (see Figure 3). These features provide the seed of the description language which will be used to define initial and goal states as well as the intermediate states during the application of SP-GEN. The resulting domain ontology is very useful, because experts naturally use the terminology at the most useful level (Rosch, 1978; Chi, Feltovich & Glaser, 1981).†

The next step on route to a formal knowledge base is to assimilate each knowledge unit to the model of expertise (i.e. the model of skeletal plan refinement), which structures the knowledge of mechanical engineering. The model distinguishes between knowledge about the product, the production plan and the environment of the company. In order to assign each term to the corresponding "module" of the knowledge base different *views* are defined: product, plan and

† In their work on DEDAL, Baudin, Kedar and Pell (1994) found that the concept definition phase is the activity that has the greatest impact on the performance of the system in the early stages of its development. The functionality of such a concept definition can be described as a form of interactive induction (Buntine & Stirling, 1990).

environment views. Combined views were introduced for representing interdependencies among two or more single views. For example, *interdependence between product and plan view* is called a combined view, which is appropriate for all features which relate properties of the workpiece to properties of the production plan. Since the views were introduced for modularizing the domain knowledge, combined views should be scarcely used.

The expert has to assign each failure to one of the defined views. For example, the feature "workpiece material alloyed steel" is assigned to the *product view.* On the other hand, "insertion directly beside the cape" is assigned to the *combined view: interdependence between product and plan.* Figure 3 illustrates how these view assignments decompose the domain ontology of features (see Phase III).

4.4. PHASE IV: CONSISTENCY AND REDUNDANCY ASSESSMENT

In the fourth phase the different knowledge units are compared. Through different consistency and redundancy checks, contradictions and redundant definitions are eliminated. CECoS supports the domain expert by detecting potential contradictions and redundant definitions based on syntactic similarities. Truly semantic discrepancies have to be found by the expert. Features of different views cannot contradict each other and cannot be redundant, because they refer to different entities. Only the interdependencies between various views (combined views) may result in contradictions and redundancies between different reference domains. Each test is applied pathwise, for single classes and for common successor classes. For example, an inconsistency may arise when a feature and its negation are attributed on the same path. This would be a classical contradiction to the subsumption principle. A redundancy may occur when the same feature is used to define a class as well as one of its successor classes. This feature should only be assigned to the superordinate class. Phase IV of Figure 3 illustrates this situation with the feature "mold not cylindrical".

4.5. PHASE V: STEPWISE FORMALIZATION

Within the last phase the informal definitions of the different problem classes are stepwise formalized. In the semiformalization step, the expert explains each feature with more detailed expressions. Thereafter, these natural language expressions are completely formalized in TAXON. Since the features are modularized according to views, this stepwise formalization process is performed view by view.

During the semiformalization step, the features are independently prepared for formalization. CECoS presents a feature together with its assigned views as a template. The expert is requested to explain this feature in more concrete terms. Thereby, new features may be generated and assigned to views as well. For the new features, the view assignment, consistency and redundancy assessment as well as the semiformalization phases are iterated, until the most detailed explanations are supplied. The most concrete features are termed *atomic.* The decisions as to which features are considered as atomic is left to the user.

The semiformalization allows for an easier transformation of the problem class descriptions into the formal representation language TAXON. The definition of the T-box is directly determined by the semiformalization. Each utilized feature forms a concept, and the concept definition is given by the more detailed explanation.

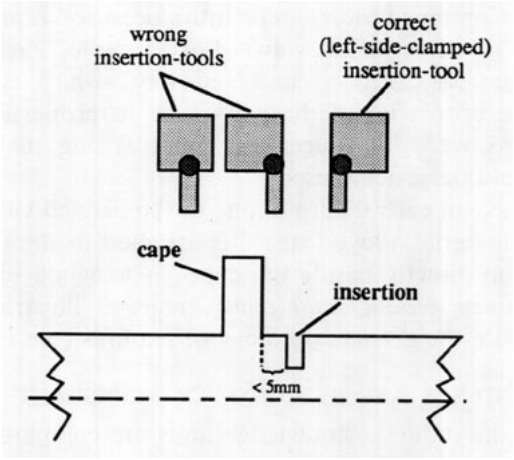

FIGURE 4. The feature "insertion directly beside the cape" denotes an interdependence between the geometry of the workpiece (i.e. the insertion) and a respective insertion tool (i.e. left-side clamped) and is therefore assigned to the combined view *interdependence between product and plan view*.

Atomic features are *primitive concepts*, whereas non-atomic features are *compound concepts* defined by the formalizations of their explaining features.

For example, consider the feature "insertion directly beside the cape" of problem class i, which has been assigned to the *interdependence between product and plan view* (see Figure 3). In the semiformalization step CECoS presents the template "insertion directly beside the cape(product and plan view) = ⟨_____⟩" The domain expert explains this feature by "*The distance between insertion and cape is less than 5 mm, and therefore a left-side clamped insertion tool is necessary.*" He points out that a special cutting tool is necessary to produce the insertion directly beside the cape. This explanation is illustrated in Figure 4 by depicting the relevant part of the respective case. Figure 4 shows two non-applicable tools as well as a suitable left-side clamped insertion tool. This explanation is then refined, until only atomic features occur.

During formalization step the feature "insertion directly beside the cape" and its explanation is defined as a compound concept in TAXON:†

```
(conc insertion_directly_beside_the_cape
  and (some i   insertion)
      (some c   cape)
      ((    (-  (i  left_border)
                (c  right_border))
            5)
      left_side_clamped_insertion_tool)).
```

"insertion", "cape", and "left_side_clamped_insertion_tool" are compound concepts, which have to be defined in more detailed terms like contour coordinates and concrete tool specifications.

The problem classes of the delineated hierarchy are formalized as concepts in

† Some technical problems have been pointed out and discussed by Schmalhofer, Reinartz and Tschaitschian (in press).

TAXON as well. For instance, Figure 3 (see phase V) shows the TAXON concept of problem class i. This concept is a conjunction of the superordinate class e and additional feature concepts. In order to define this class, these features must be formalized as (primitive or compound) concepts as has already been illustrated for the feature "insertion directly beside the cape".

The KL-ONE-like inference services (such as classification of the taxonomy, subsumption queries, etc.) enable CECoS to check the correctness of the formalized problem class hierarchy and to solve the classification task for newly arising production problems.

4.6. CLASSIFICATION OF NEWLY ARISING PRODUCTION PROBLEMS

After the whole problem class hiearchy is formalized, the TAXON inference system classifies known and newly arising production problems, which are represented in adequate terms. Figure 5 again depicts the subtree of problem class 5. The initial exemplar cases are classified as members of their respective terminal problem class. Assume a new production problem, which is defined by A-box assertions outlined at the lower left side of Figure 5. When TAXON classifies a newly arising production problem, a defining feature like "mold_cylindrical" is compared to the top of the hierarchy. The inference mechanisms should enable the system to test whether the detailed description corresponds to a crucial feature. If each feature of a problem class definition is successfully proved, the procedure

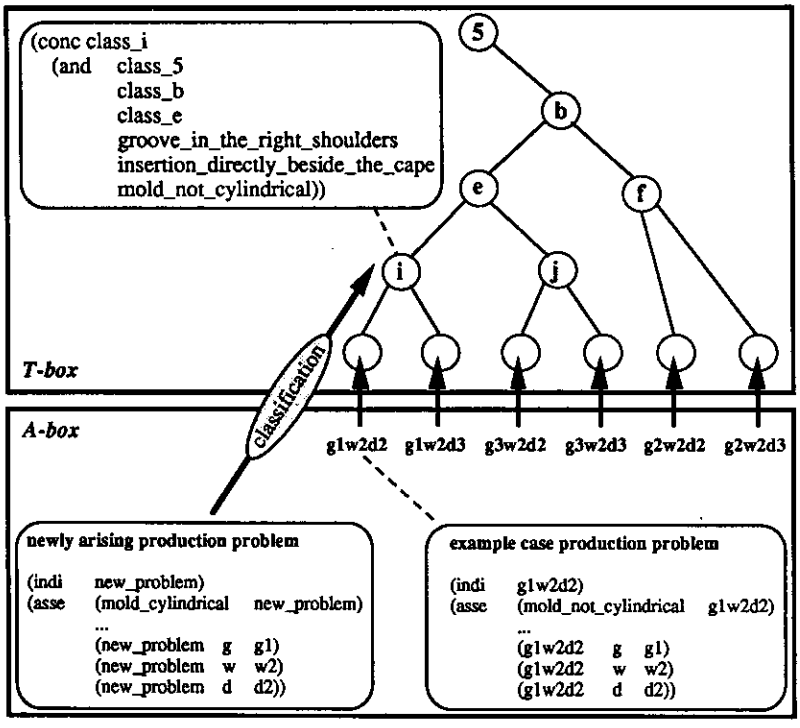

FIGURE 5. The TAXON inference system classifies a newly-arising production problem into the most specific problem class.

continues with its successor classes. This process is iterated until a mismatch occurs or a terminal class is obtained (compare with Aamodt, 1991; Porter *et al.*, 1990). In our case, the "new_problem" passed all the tests until problem class i is reached. For problem class i a mismatch occurs, because of the defining feature "mold_cylindrical". Since a mismatch is also obtained for problem class j, the "new_problem" is categorized into class e. This classification is a prerequisite for retrieving a skeletal plan. Each skeletal plan is attached to a specific problem class. How such skeletal plans are obtained by the explanation-based learning procedure SP-GEN will be described next.

5. Explanation-based learning to form generalizations

SP-GEN is based on the idea of explanation-based learning (Mitchell *et al.*, 1986), which is a form of analytic learning. Explanation-based learning (EBL) is a method where generalizations are formed from a single case by employing a body of background knowledge, a target concept and a desired description level, which is defined by a so called operationality criterion. EBL explains or proves that the case (*training example*) satisfies the requirements of the target concept (*goal concept*) using the background knowledge as the *domain theory*. In other words it is deduced from the *domain theory* that the *training example* is an instance of the *goal concept*. The deduced proof is generalized to a more usable concept definition in terms of the desired description level (*operationality criterion*). This generalization is guided by the background knowledge rather than by some unknown inductive bias and brings the relevant features of the case in focus while irrelevant features are ignored.

A characterization of SP-GEN as an EBL-method is presented in Table 1. The domain theory provides significantly more structure than the domain theories of similar systems (DeJong & Mooney, 1986; Minton, Carbonell, Etzioni, Knoblock & Kuokka, 1987; Mitchell *et al.*, 1986). Unlike DeJong & Mooney's GENESIS that uses a goal state as a target concept a generic problem and solution description is used. A specific manufacturing problem and its solution serves as training example. Different operationality criteria are used for different levels of the problem class hierarchy. The learning result of SP-GEN is a blueprint (i.e. a skeletal plan) that can be reused for manufacturing similar workpieces.

TABLE 1
Characterization of SP-GEN as an EBL-method

Domain theory	Hierarchically structured knowledge base in terms of state and operator definitions, problem classes, abstraction and refinement rules
Target concept	Generic problem and solution approach (non-operational)
Training example	Success case: initial and goal states, sequences of concrete operations (e.g. a production plan that has been successfully used in the real world)
Operationality criteria	Terms used by domain experts to describe states and operators (acquired with CECoS and COKAM+)
Learning result	A skeletal plan as a blueprint for solving future problems (i.e. a well structured specification of a problem description and its solution in operational terms)

The complexity of the application domain required a decomposition in several processing phases. A skeletal plan is thus constructed in four phases. In the first phase, the execution of the source plan is simulated and explanations for the effects of the individual operations are constructed. In the second phase the generalization of these explanations is performed with respect to a criterion of operationality that specifies the vocabulary for defining general operators for the skeletal plan. In the third phase, a dependency analysis of the resulting operator effects unveils the substantial interactions of the plan at the more general level of the skeletal plan. And finally in the fourth phase the concept descriptions for the generalized operators of the skeletal plan are formed by collecting and normalizing the important constraints for each operation that were indicated by the dependencies.

Although TAXON could in principle be used to represent production plans, a PROLOG-like notation has actually been used. Since approaches to translating terminological representations into a kind of modal logic and then using resolution like inference methods, which are easy to realize in PROLOG, already exist (Ohlbach, 1991), the two representation formalisms can be combined in a straightforward manner.

The formal representation of a specific operation (cut 4) from the concrete plan of problem case g5w1d3 (see also Figure 1) is shown by the following example:

```
case("g5w1d3", plan("cut 4"),
  cut(speed(400),
      feed(0·20),
      (path([polygon([5000, 300), (4500, 300), (3900, 400)]),
             polygon([(3050, 400), (3000, 410), (2760, 410),
                      (2750, 450)]),
             polygon([(1470, 450), (1400, 500), (650, 500),
                      (600, 525)])]),
      tool(toolholder("ISO_5608"("CS-NL3240-12")),
           insert("ISO_1832"("SNGN120808T02020"),
           "SH20F")))).
```

5.1. SIMULATION AND EXPLANATION

In the first phase of SP-GEN, the plan execution is simulated on the basis of the available domain theory. The simulation of the plan is performed by sequentially determining the effects of each operator $\mathbf{Op}_1, \ldots, \mathbf{Op}_n$ of the plan. In order to determine the effects of the sequence of operators, the intermediate preocessing states from the initial state $\mathbf{S}_0$ (the mold) to the final state $\mathbf{S}_n$ (which will contain the target workpiece if the domain theory is sufficient) are computed as follows:

$$\mathbf{S}_0 \xrightarrow{\mathbf{Op}_1} \mathbf{S}_1 \xrightarrow{\mathbf{Op}_2} \mathbf{S}_2 \cdots \mathbf{S}_{n-1} \xrightarrow{\mathbf{Op}_n} \mathbf{S}_n$$

The effects of each operator are represented by a set of rules with STRIPS like add and delete actions. The execution of these rules thus creates the successor world state. By applying all the rules for each operator, the various consequences of the

individual operations of the plan are calculated. The proofs that exist for applicability of each operator rule can now be seen as an explanation of each eff that depends on operator attributes as well as world state attributes, from the init or intermediate states. The following example shows the explanation of cut 4 in t context of case g5w1d3:

```
"cut 4" is a very_fine_cut
Testing precondition: chucking_precision_sufficient
Testing precondition: is_chucked
Testing precondition: within_workspace_boundaries
Testing precondition: no_collision_with_workpiece
Testing precondition: cutting_parameters_ok
Testing precondition: machine_rpm_sufficient
All preconditions successfully tested.
  very_fine_cut since
    high_tolerance_required and
    Chucking_precision(very_high) and
    is_chucked and
    within_workspace_boundaries and
    cutting_path([polygon([5000, 300), . . .])]) and
    cutting_direction(tool(· · ·)), to_left) and
    path_direction([polygon([(5000, 300), . . .])], to_left) and
    max_tool_cutting_depth(tool(· · ·), 80) and
    max_cutting_depth([polygon([5000, 300), . . .])], 5) and
    80 > 5 and
    cutting_material(ceramic) and
    workpiece_material(cast_iron) and
    cutting_feed(00·2) and
    400 < 800 and
    00·2 < 00·8 and
    min_x_coordinate(300) and
    circumference(300, 00·1884) and
    machine_rpm(5000) and
    cutting_speed(400) and
    400 < 00·1884 * 5000.

New Surface: rsr(5000, 50, 300,—)
New Surface: rscj(5000, 4500, 300, —)
New Surface: rsec(4500, 3900, 300, 400, —)
. . .
New Surface: rsec(650, 600, 500, 525, —)
```

If the domain theory is sufficient, a complete explanation for the whole plan will be obtained. Otherwise, a specific gap has been identified in the domain theory. CECoS and COKAM+ can then be used to fill this knowledge gap.

5.2. GENERALIZATION

In the second phase of the procedure, these proofs are independently generalized for each production step of the plan. The independent generalization of each production step is necessitated because of the complexity of the complete plans.

The degree of generalization is determined by the operationality criteria for each production step, which are defined at the concept (Hirsh, 1990) rather than at the predicate level. These criteria are obtained from the terms which the expert used for describing the different operations of the concrete plan at a general level. It is thus assumed that exactly those terms which are used by experienced humans would determine operationality. A justification for this assumption can be found in the research of Rosch (1978) who has shown that humans favour basic level categories in their descriptions. Such categories can be termed operational in the sense that they provide maximum information and the least cognitive effort for achieving some task goal. The following example shows the generalized proof for cut 4 in the context of case g5w1d3:

very_fine_cut with
  high_tolerance_required
  cutting_path(_g30125)
  cutting_tool(_g30135)
  cutting_direction (_g30135, _g30147)
  path_direction(_g30125, _g30147)
  max_tool_cutting_depth(_g30135, _g30171)
  max_cutting_depth(_g30125, _g30183)
  _g30171 > _g30183
  cutting_material(ceramic)
  cutting_feed(_g30225)
  _g30235 < 800
  _g30225 < 00·8
  min_x_coordinate(_g30259)
  circumference(_g30259, _g30271)
  cutting_speed(_g30235)
  _g30235 < _g30271 * _g30281

5.3. DEPENDENCY ANALYSIS

The dependency analysis of the third phase determines which previous operations (or initial state affairs) achieved the prerequisites for the various production steps of the plan. It is thereby determined when the prerequisites for performing a specific production step were accomplished. A directed graph is constructed in which all existing dependencies between the individual plan operations and the problem description are denoted by arcs (compare with Stefik, 1981). The following example shows selected dependencies for cut 4 in the context of case g5w1d3:

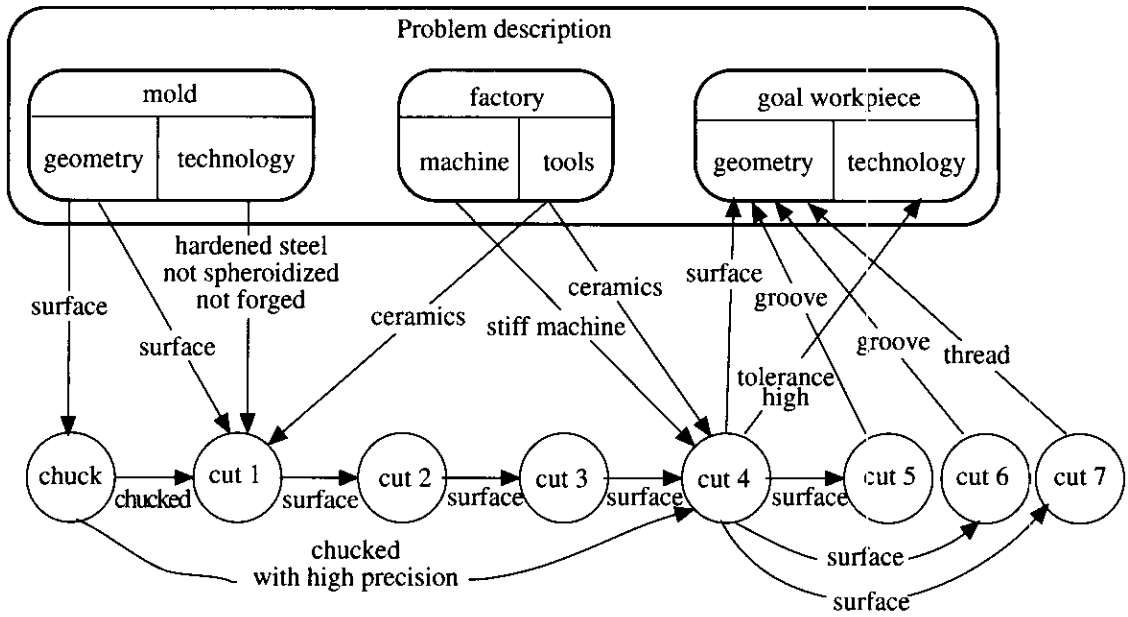

FIGURE 6. Partial dependency graph for the case g5w1d3.

"cut 4" produces rsec(650, 600, 500, 525, —)
required for the goal workpiece.
"cut 4" produces rsec(2760, 2750, 410, 450, —)
required for "cut 6".
"cut 4" produces rsec(3050, 3000, 400, 410, —)
required for "cut 5".
"chuck" produces chucking_precision (very_high)
required for "cut 4".
"cut 4" requires machine_stiffness (very good)
given in the environment description.
"cut 4" requires workpiece_material (cast_iron)
given in the workpiece description.

For the complete case g5w1d3, Figure 6 shows a graphical representation of selected dependencies. For example, cut 1 depends on the workpiece being chucked (for subsequent cuts this obvious dependency is no longer shown in the graph), on the geometry and the technology of the mold, and on the availability of ceramic cutting tools in the factory. The required geometry and technology of the goal workpiece is produced exclusively by cuts 4 to 7. The lack of a dependency between cuts 5 to 7, furthermore indicates that they could be executed in any sequence.

5.4. NORMALIZATION

This last phase builds the skeletal plan in its final representation by identifying independently solvable sub-formulas from the dependency graph which expresses only local constraints on one operator. By analysing the occurrence of variables in the graph the dependencies are separated into:

- one set $\mathbf{R}_{\text{Enable}}$ containing all dependencies which only relate to features of the problem description,
- one set $\mathbf{R}_{\mathbf{Op}_i}$ for each operator $\mathbf{Op}_i$ where the dependencies refer to parameters of the operator $\mathbf{Op}_i$,
- one set $\mathbf{R}_{\text{Dependent}}$ where the dependencies refer to the possible orderings of the operator classes.

The set of constraints $\mathbf{R}_{\text{Enable}}$ formally describes the class of problems for which the skeletal plan can be used: it specifies the application conditions for the skeletal plan. The skeletal plan itself consists of the set of operator classes $\mathbf{Op}_1, \ldots, \mathbf{Op}_n$ with the constraints $\mathbf{R}_{\mathbf{Op}_i}$ and $\mathbf{R}_{\text{Dependent}}$ which specify the possible sequences in which they may be applied. Some of the application conditions for the skeletal plan acquired from the concrete production problem g5w1d3 are:

- long_workpiece
- left_vertical_plane
- right_center_hole
- workpiece_material (cast_iron)
- machine_stiffness (very_good)

6. Implementation and evaluation

For testing the proposed integration of knowledge acquisition and machine learning techniques in the COEX architecture, CECoS and SP-GEN have been implemented in PROLOG (compare with Kedar-Cabelli & McCarty, 1987). CECoS runs on a MAC II and SP-GEN runs on a MAC II as well as on SPARC work stations.

The *paired comparison* and *hierarchical cluster analysis* of CECoS have been tested with 60 production problems and resulted in the problem classes which are partially shown in Figure 3 (lower section). These components worked very well in that it required little time to establish the problem class hierarchies and that the results were readily validated by the human expert. Since the *feature definition* requires more user interactions from the human expert, this component was only applied to a selected subset of problem classes. Although it was often difficult for the expert to name appropriate features, a complete description of each class was obtained after a few iterations. Several features were assigned to a combined view within the *view assignment* phase. This shows that there are local interdependencies between the meta classes of the KADS-model, which are not represented in the model itself. Nevertheless, views partition the knowledge into mostly independent segments, and the *consistency and redundancy assessment* could thus be independently performed for these segments. For the *semiformalization,* templates were automatically generated and filled by the expert. Although the *formalization* of such restricted natural language expressions can be performed largely automatically (Schmidt & Wetter, 1989; Schmalhofer & Schmidt, 1991), we have not yet integrated such a facility and conducted the formalization by hand. The resulting knowledge base gives a formal description of the problem classes exemplified for class 5 in Figure 3.

SP-GEN has been applied for constructing the skeletal plan for one of the terminal classes. From this demonstration it could be inferred that SP-GEN will

generally work for all terminal classes, i.e. it can perform any kind of instance to class generalization.

7. Discussion

The field of machine learning has accomplished tremendous achievements during the last decade. One particularly active area is the research on knowledge-intensive learning mechanisms and more specifically explanation-based learning. Explanation-based learning is a very promising learning technique, because it enables the learning biases to be made explicit in the form of domain theories (Mitchell *et al.*, 1986). Moreover, under certain conditions it may be used for speed-up learning (e.g. Mooney, 1989; Minton, 1990).

With EBL, one can generalize training examples under the guidance of a target concept and a domain theory. An operationality criterion determines the degree of generalization. Initially, the application of EBL required a complete and consistent domain theory, which is sufficient to prove that a training example belongs to the target concept. More recent EBL systems may even be applied without a complete domain theory (e.g. Duval, 1991). Other extensions concern the operationality criterion. Instead of a static operationality criterion, dynamic reasoning process may be used to determine (conditional) operationality (DeJong & Mooney, 1986; Hirsh, 1990). The first approaches to EBL did not generalize the structure of the explanation, but relaxed constraints on the variables in the explanation of a specific example. Shavlik (1990) showed how recursive and iterative concepts can be learned by generalizing the explanation structure. The progress in machine learning is well documented by empirical performance evaluations of different algorithms in various well structured microworlds (e.g. the blocks world, the Tower of Hanoi task, simple sorting tasks.).

All this progress in machine learning and EBL does not pay off, unless the machine learning community can present success cases, where these techniques are applied to real world problems. The development of knowledge-based systems for complex tasks like production planning in mechanical engineering is a complex application in which economic returns could be achieved with EBL. Therefore, we tried to adapt the EBL paradigm to this real world application domain.

In pursuing this goal, a number of issues had to be addressed which usually do not arise in microworlds: how can we get the relevant background knowledge and establish an adequate domain theory? How should one determine operationality in real world domains? How can we ensure the utility of the formed generalizations? How should the training examples be selected? These are just a few of the questions which need to be answered before EBL can be applied to the real world.

We found the apprenticeship learning paradigm a useful framework for addressing these questions. In the apprenticeship learning paradigm, an expert (or human user) and automatic machine learning techniques cooperate to construct a knowledge-based system. Thereby, the expert and the system learn from each other. Through knowledge elicitation, the system can acquire new knowledge from the expert. The experts, on the other hand, can learn how their knowledge must be structured, so that it can be documented and communicated by a knowledge-based system (Schmalhofer, Reinartz & Tschaitschian, 1992). This is the framework that was used

for developing the COEX architecture, where model-based knowledge engineering techniques, knowledge elicitation and conceptual clustering were integrated with EBL. CECoS is an implementation of conceptual clustering, and SP-GEN is an implementation of EBL within this framework.

The COEX system is thus a type of apprenticeship learning system. The expert takes the role of an author and provides new background knowledge with respect to selected success cases from the real world. This activity is supported by the tools CECoS and COKAM+ (Schmidt, 1992). The system takes the role of an editorial assistant and produces representations at the appropriate level of generality (supported by SP-GEN) and gives feedback on whether the information provided by the expert is comprehensive in terms of the available representation formalisms (supported by CECoS).

The results of our implementations show the usefulness of the KA/EBL integration for complex domains. CECoS forms hierarchically structured problem classes so that the so called utility problem can be handled (Minton, 1990). SP-GEN was successful in forming skeletal plans for the terminal problem classes. SP-GEN generalizes constants such as feed $0{\cdot}45\,\mathrm{mm\,U^{-1}}$ into variables (Feed1), which are constrained ($3 < \mathrm{Feed1} < 5$) according to a dependency or validation structure for the whole plan (Kambhampati, 1990). It was thus shown that with the proposed integration scheme, EBL can be applied to real world problems.

We also encountered some problems with CECoS as well as with SP-GEN, which could not have been unveiled with toy applications. The application of CECoS suffered from the representation limitations of the terminological language TAXON. Because of these limitations, a feature such as "groove_in_the_right_shoulders" could not be represented. This feature means that there would be a sequence of shoulders at the right side of the cape and that there would be a groove in this area. Whether there would be two, three or more shoulders is not essential and must consequently remain undertermined. These representation problems are discussed by Schmalhofer, Reinartz and Tschaitschian (in press).

Secondly, when we tried to apply SP-GEN to non-terminal problem classes, it became clear that explanation-based generalization is not sufficient for these classes. Whereas SP-GEN can form generalizations from cases for quite specific problem classes, true abstractions would be required to generate procedure schemata for the more abstract problem classes. Knoblock (1990) addresses the issue of abstraction for problem solving. Michalski and Kodratoff (1990) have recently pointed out that generalization needs to be distinguished more clearly from abstraction. While generalization normally uses the same representation language, abstraction involves a change in the representation space to transform the representation language into a simpler language than the original. The problem descriptions of the non-terminal classes were at such an abstract description level.

In order to generate a procedure schema for these classes, one must combine different numbers and different sequences of operators. But SP-GEN offers no support to generalize over the cutting sequence. Although the dependency analysis of SP-GEN finds interdependencies between single cutting steps, it cannot construct a uniform procedure schema, that subsumes substantially different operator sequences and plans with different numbers of operators. The construction of such procedure schemata requires the formation of true abstractions, which led us to

develop the explanation-based abstraction method and the PABS procedure (Schmalhofer & Tschaitschian, 1993). With PABS the proposed KA/EBL integration will become even more useful and may provide a significant milestone to a unified approach to learning in complex domains (Schmalhofer *et al.*, in press).

This research was supported by grants ITW 8902 C4 and 413-5839-ITW9304/3 from the BMFT and by grant SCHM 648/1 from the DFG. We thank all previous and current colleagues in the ARC-TEC, KIWi and VEGA projects who contributed to this research in one way or another. More specifically, we would like to acknowledge the contributions of Bidjan Tschaitschian and Otto Kühn to the development of the tools CECoS and SP-GEN. Comments by Gheorghe Tecuci, Smadar Kedar and Yves Kodratoff helped to substantially improve the readability of the paper and are much appreciated.

References

AAMODT, A. (1991). *A knowledge-intensive, integrated approach to problem solving and sustained learning.* Doctoral Dissertation, University of Trondheim, Norway.

AITKEN, S. KÜHN, O., SHADBOLT, N. & SCHMALHOFER, F. (1993). A conceptual model of hierarchical skeletal planning and its formalization, In LÖCKENHOFF, G. Ed. *Proceedings 3rd KADS User Meeting,* Munich, Germany: Siemens AG.

BACKHAUS, K., ERICHSON, B., PLINKE, W., SCHUCHHARD-FICHER, C. & WEIBER, R. (1987). *Multivariate Analysemethoden.* Springer: Berlin.

BAUDIN, C., KEDAR, S. & PELL, B. (1994). Increasing levels of assistance in refinement of knowledge-based retrieval systems. *Knowledge Acquisition,* **6,** 179–196.

BOOSE, J. H. & GAINES, B. R. (1989). Knowledge acquisition for knowledge-based systems: notes on the state-of-the-art. *Machine Learning,* **4,** 377–394.

BRACHMAN, R. J. & SCHMOLZE, J. G. (1985). An overview of the KL-ONE knowledge representation system, *Cognitive Science,* **9,** 171–216.

BREUKER, J. & WIELINGA, B. (1989). Models of expertise in knowledge acquisition, In GUIDA, G. & TASSO, C. Eds. *Topics in expert system design, methodologies and tools,* pp. 265–295. Amsterdam: North-Holland.

BUNTINE, W. & STIRLING, D. (1990). Interactive induction, In: HAYES, J. MICHIE, D. and TYUOGO, E. Eds. *Machine Intelligence 12, Analysis and synthesis of knowledge,* pp. 121–137. New York: Oxford University Press.

BURSTEIN, M. H. (1983). Concept formation by incremental analogical reasoning and debugging, In MICHALSKI, R. S., CARBONELL, J. G. and MITCHELL, T. M. Eds. *Machine Learning, Vol. II,* pp. 351–370. Palo Alto: Tioga.

CHI, M. T. H., FELTOVICH, P. J. & GLASER, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science,* **5,** 121–152.

DEJONG & MOONEY, R. (1986). Explanation-based learning: an alternative view. *Machine Learning,* **1,** 145–176.

DUVAL, B. (1991). Abduction for explanation-based learning, In: KODRATOFF, Y. *Lecture notes in AI: Machine Learning—EWSL-91,* PP. 348–360. Berlin: Springer.

FELDMÜHLE, A. G. (1984). *SPK: Examples for application (turning and milling),* D-7310 Plochingen Neckar, Germany: Feldmühle AG, Produktbereich SPK-Werkzeuge.

FISHER, D. H. (1987). Knowledge acquisition via incremental conceptual clustering, *Machine Learning,* **2,** 139–172.

HANSCHKE, P., ABECKER, A. & DROLLINGER, D. (1991). TAXON: a concept language with concrete domains. *Proc. International Workshop on Processing Declarative Knowledge (PDK '91),* pp. 411–413, 1–3 July. Kaiserslautern, Germany.

HIRSH, H. (2990). Conditional operationality and explanation-based generalization, In KODRATOFF, Y. & MICHALSKI, R. S. Eds. *Machine Learning: an artificial intelligence approach.* Vol. 3, pp. 383–395. San Mateo, CA: Morgan Kaufmann.

KAMBHAMPATI, S. (1990). Mapping and retrieval during plan reuse: a validation structure

based approach. *Proc. Eighth National Conference on Artificial Intelligence (AAAI '90)*, pp. 170–175. Landham, MA: MIT Press.

KEDAR-CABELLI, S. T. & MCCARTY, L. T. (1987). Explanation-based generalization as resolution theorem proving. *Proc. Fourth International Workshop on Machine Learning*, University of California, Irvine, pp. 383–389.

KNOBLOCK, C. A. 1990. Learning Abstraction Hierarchies for Problem Solving. *Proc. Eighth National Conference on Artificial Intelligence*, Boston, MA, pp. 923–928.

KODRATOFF, Y. & TECUCI, G. (1987). Techniques of design and DISCIPLE learning apprentice, *International Journal of Expert Systems*, **1**, 39–66.

KÜHN, O. & SCHMALHOFER, F. (1992). Hierarchical skeletal plan refinement task- and inference structures, In BAUER, C. & KARBACH, W. Eds. *Proc. 2nd KADS User Meeting*. Munich, Germany: Siemens AG.

LEBOWITZ, M. (1986). Concept in a rich input domain: generalization-based memory, In: MICHALSKI, R. S., CARBONELL, J. G. & MITCHELL, T. M. Eds. *Machine Learning, Vol. II*, pp. 193–214. Palo Alto: Tioga.

LEBOWITZ, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, **2**, 103–138.

MAAREK, Y. S. (1990). An incremental conceptual clustering algorithm that reduces input-ordering bias, In: GOLUMBIC, M. C. Ed. *Advances in Artificial Intelligence*, pp. 129–144. Berlin: Springer-Verlag.

MICHALSKI, R. S. & KODRATOFF, Y. (1990). Research in machine learning: recent progress, classification of methods, and future directions, In: KODRATOFF, Y. & MICHALSKI, R. S. Eds. *Machine Learning: an artificial intelligence approach*, Vol. 3, pp. 3–30. San Mateo, CA: Morgan Kaufmann.

MICHALSKI, R. S. & STEPP, R. E. (1983). Automated construction of classifications: conceptual clustering versus numerical taxonomy, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. **Pami-5**, 396–410.

MINTON, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, **42**, 363–392.

MINTON, S., CARBONELL, J. G., ETZION, O., KNOBLOCK, C. A. & KUOKKA, D. R. (1987). Acquiring effective search control rules: explanation-based learning in the PRODIGY system. *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 122–133. Irvine, CA: Morgan Kaufmann.

MITCHELL, T. M., KELLER, R. M. & KEDAR-CABELLI, S. T. (1986). Explanation-based generalization: a unifying view, *Machine Learning*, **1**, 47–80.

MITCHELL, T., MAHADEVAN, S. & STEINBERG, L. (1985). LEAP: a learning apprentice system for VLSI design, In *Proc. 9th International Joint Conference on Artificial Intelligence*, Los Angeles, CA: Morgan Kaufmann. pp. 573–580.

MOONEY, R. J. (1989). The effect of rule use on the utility of explanation-based learning, In: *Proc. Eleventh International Joint Conference on Artificial Intelligence*, Los Angeles, CA: Morgan Kaufmann. pp. 725–730.

OHLBACH, H. J. (1991). Semantics based translation methods for modal logics, *Journal of Logic and Computation*, **1**, 691–746.

PORTER, B. W., BARREIS, R. & HOLTE, R. C. (1990). Concept learning and heuristic classification in weak-theory domains, *Artificial Intelligence*, **45**, 229–263.

REINARTZ, T. (1991). *Definition von Problemklassen im Maschinenbau als eine Begriffsbildungsaufgabe*, German Research Center for Artificial Intelligence, Document, No. 91-18. Kaiserslautern.

ROSCH, E. (1978). Principles in categorization, In: ROSCH, E. & LLOYD, B. Eds. *Cognition and categorization*, pp. 27–71. Hillsdale, NJ: Lawrence Erlbaum.

SCHMALHOFER, F., BERGMANN, R., KÜHN, O. & SCHMIDT, G. (1991). Using an integrated knowledge acquisition method to prepare sophisticated expert plans for the reuse in novel situations, In: CHRISTALLER, T. Ed. *Proc. 15th German Workshop on Artificial Intelligence*, Berlin: Springer-Verlag, pp. 62–71.

SCHMALHOFER, F., KÜHN, P. & SCHMIDT, G. (1991). Integrated knowledge acquisition from text, previously solved cases, and expert memories, *Applied Artificial Intelligence*, **5**, 311–337.

SCHMALHOFER, F., REINARTZ, T. & TSCHAITSCHIAN, B. (1992). Intelligent documentation as a catalyst for developing cooperative knowledge-based systems, In: WETTER, T., ALTHOFF, K.-D., BOOSE, J., GAINES, B. R., LINSTER, M. & SCHMALHOFER, F. Eds. *Current developments in knowledge acquisition—EKAW '92,* Berlin: Springer-Verlag.

SCHMALHOFER, F., REINARTZ, T. & TSCHAITSCHIAN, B. A unified approach to learning for complex real world domains. *Applied Artificial Intelligence,* Vol. 9 (in press).

SCHMALHOFER, F. & SCHMIDT, G. (1991). Situated text-analysis with COKAM+, In: WOODWARD, B. Ed. *Sisyphus Working Papers: EKAW '91,* Crieff, UK, pp. 24–34.

SCHMALHOFER, F. & THOBEN, J. (1992). The model-based construction of a case-oriented expert system, *AI Communications,* **5,** 3–18.

SCHMALHOFER, F. & TSCHAITSCHIAN, B. (1993). The acquisition of a procedure schema from text and experiences. *Proc. 15th Annual Confernce of the Cognitive Science Society,* pp. 883–888, Hillsdale, NJ: Lawrence Erlbaum.

SCHMIDT, G. (1992). Knowledge acquisition from text in a complex domain. *Proc. Fifth International conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems,* pp. 529–538, 9–12 June. Paderborn, Germany. Berlin: Springer-Verlag.

SCHMIDT, G. & WETTER, T. (1989). Towards knowledge acquisition in natural language dialogue, In: *Proc. 3rd European Workshop on Knowledge Acquisition for Knowledge-Based Systems.*

SHAVLIK, J. W. (1990). Acquiring recursive and iterative concepts with explanation-based learning. *Machine Learning,* **5,** 39–70.

STEFIK, M. (1981). Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence,* **16,** 111–140.

TECUCI, G. & KODRATOFF, Y. (1990). Apprenticeship learning in imperfect domain theories, In: KODRATOFF, Y. and MICHALSKI, R. S. Eds. *Machine Learning: an artificial intelligence approach.* Vol. 3, pp. 514–551. San Mateo, CA: Morgan Kaufmann.

TSCHAITSCHIAN, B. (1991). *Eine integrative Wissenserhebung mit CECoS: Konzepte und prototypische implementierung,* Universität Kaiserslautern. Projektarbeit im Fachbereich Informatik.