KNACK: A Knowledge Acquisition Tool for Systems that Evaluate Designs

Georg Klinker
Department of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

KNACK is a specialized knowledge acquisition tool that generates expert systems for evaluating different classes of designs. The tool derives its power from exploiting the presupposed problem-solving method of the expert systems it generates. An important feature of KNACK is that it acquires knowledge from domain experts without presupposing knowledge engineering skills on their part. This is achieved by incorporating general knowledge about evaluation tasks in KNACK. Using that knowledge, KNACK builds a model of the domain through an interview process with the expert. During knowledge acquisition KNACK uses the domain model to elicit knowledge in a format familiar to the expert. KNACK expects the expert to communicate a portion of his knowledge as a sample report and divides the report into small fragments. It asks the expert for strategies of how to customize the fragments for different applications. KNACK generalizes the fragments and strategies, displays several instantiations of them, and the expert edits any of these that need it. The corrections motivate and guide KNACK in refining the knowledge base. This process of abstraction and completion results in a knowledge base containing a large collection of generalized report fragments more broadly applicable than the sample report. Finally, KNACK examines the acquired knowledge for incompleteness and inconsistency.[1]

1. Introduction

KNACK [Klinker 87a] is a knowledge acquisition tool that can be used by domain experts to create expert systems that assist with the evaluation of different classes of designs. It exploits a presupposed problem-solving method as well as an explicit domain model and it takes a report-driven approach to acquire knowledge.

KNACK presupposes and exploits the problem-solving method of the expert systems it generates. A problem solving method is knowledge that establishes and controls the sequences of actions required to

[1]This research was sponsored by the Defense Nuclear Agency (DNA) and the Harry Diamond Laboratories (HDL) under contract DNA001-85-C-0027, the Bares Foundation, Boeing Computer Services (BCS), and Digital Equipment Corporation (DEC). The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of DNA, HDL, the Bares Foundation, BCS, or DEC.

perform some task. This control knowledge dynamically defines the order in which subtasks have to be solved in order to perform the overall task. It also defines the kind of domain specific knowledge that is applicable within each step. Thus, the problem solving method helps to identify and classify the domain knowledge. It makes the different roles knowledge plays in the design evaluation task explicit and suggests ways to organize the knowledge base according to the knowledge roles. It further defines how knowledge interacts during problem solving. The granularity of the problem solving method is determined by the demand that the knowledge represented by a knowledge role can be applied without further control knowledge, e.g. the order in which that knowledge will be brought to bear does not matter.

Like SALT [Marcus 87], KNACK can be used to develop expert systems that construct a solution compatible with a set of constraints. But whereas SALT-generated expert systems produce designs from scratch, i.e., typically one designer has complete knowledge about all constraints a solution has to satisfy, KNACK generates evaluation systems. Evaluation systems assume that multiple designers are involved in a design task and each designer only knows a subset of the constraints a solution has to satisfy. The designers have to work together to construct a design. They must communicate their partial solution to other designers who then refine the design. Evaluation systems are useful when designers have little or no knowledge about specialized design practices required by unfamiliar design techniques or environments. For example, this is the case when new techniques have been developed that are not yet common knowledge. Therefore, evaluation systems assist a designer in refining a given design to take into account aspects of an unfamiliar environment. They evaluate designs from a different point of view not anticipated in the original design. Evaluation systems assume that an incomplete design exists that may be suboptimal under a particular perspective, compare the parameters of a given design with constraints imposed by a given environment, and propose fixes in case some constraints are violated. x

Explicit knowledge about an assumed problem-solving method and its associated knowledge roles can be used during knowledge acquisition to guide a domain expert in defining, analyzing and testing a knowledge base. However, the expert is still required to enter knowledge in a structured format that may be unfamiliar to him. This imposes a learning burden on the expert. An important goal in the development of KNACK is that it acquires knowledge from domain experts without presupposing knowledge engineering skills on their part. Like OPAL [Musen 87], KNACK exploits a domain model during knowledge acquisition. The domain model contains a structural and functional definition of a particular evaluation domain. It describes the concepts experts use and their interdependencies. KNACK uses the domain model to elicit knowledge in a format familiar to the expert and develop expectations about the knowledge the expert might provide. KNACK differs from OPAL in that the domain model can be customized for a particular domain and no knowledge engineering expertise is required to build a domain model.

Another characteristic of KNACK is its report-driven approach to acquiring knowledge. KNACK assumes that an expert can present his knowledge adequately in the form of a report. The expert must have a clear understanding of what constitutes an acceptable report describing and evaluating a design. This includes that the expert knows what information is needed, how to evaluate this information, and how a designer should present this information. We think this is a valid assumption for a variety of evaluation tasks. In general, someone whose job is to evaluate the work of others must have comprehensive and precise knowledge about that work.

The following sections describe the KNACK tool in detail. Section 2 introduces the KNACK-generated expert systems and explicates their problem solving method and its knowledge roles that KNACK assumes. x

Section 3 discusses the approach KNACK takes to acquire knowledge and presents a sample interaction with KNACK. Section 4 ~~is meant to~~ shows how KNACK detects cues that its knowledge base might be incomplete or inconsistent. Section 5 describes the knowledge base KNACK generates as a result of a knowledge acquisition session. ~~In~~ Section 6 ~~we~~ introduces the systems we have built with KNACK so far. Section 7 compares KNACK to other knowledge acquisition systems.

2. The Presupposed Problem-solving Method and its Knowledge Roles

Each of the evaluation systems produced by KNACK is called a WRINGER. A WRINGER expert system assumes that an incomplete design exists that may be suboptimal under a particular perspective. Its purpose is to assist a designer in refining a given, incomplete design to take into account aspects of an unfamiliar environment. It further presents this design, together with a preliminary design evaluation, in the form of a report. To evaluate designs, a WRINGER must have available an initial description of the design to be evaluated. Thus, a WRINGER first gathers the information describing an existing design and then evaluates the information. If, as a result of the evaluation, additional information is required, a WRINGER gathers that information. After that, another evaluation is performed. This iterative process ends when the designer is satisfied with the design.

To gather the information describing a specific design a WRINGER uses strategies to elicit information from the designer or to infer it. For example, it asks questions or computes numeric values using formulas. As it progresses, the gathering of information is driven by previously elicited information. This is a data-driven approach that modifies a WRINGER's behavior according to the information specific to each design it is applied to.

The collected information is evaluated by a WRINGER for validity, consistency, completeness, and possible design flaws, i.e., a WRINGER checks the information describing a design for violations of constraints imposed by a given environment. If indications of design flaws are found, a WRINGER points them out to the designer together with suggestions for improving the design. If the designer agrees with the fix, a WRINGER updates the design description using a truth maintenance system. Finally, when the designer is satisfied with the design, a WRINGER generates a report describing and evaluating the design.

Throughout the paper we will use the Design Parameters Report WRINGER (DPR WRINGER), one of the WRINGERs we have generated with KNACK, as an example to illustrate the WRINGERs and KNACK. The domain of the DPR WRINGER is nuclear hardening. Nuclear hardening implies the use of specific engineering design practices to increase the resistance of an electromechanical system to the environmental effects generated by a nuclear weapon. Designers of electromechanical systems usually have little or no knowledge about the specialized analytical methods and engineering practices of the hardening domain. The purpose of the DPR WRINGER is to assist a designer in improving given designs of electromechanical systems that may be suboptimal from a hardening perspective. The WRINGER assumes that the initial design describes a technically functional system. It evaluates the design from a hardening perspective. The suggested improvements are either extensions to the design or recommendations for using different design components. The WRINGER presents the design, together with the results of the evaluation, in the form of a technical document that meets government requirements.

In detail, the DPR WRINGER documents a system description, analysis, design features, and assumptions required to assure the nuclear hardness and survivability of a system with respect to one nuclear

More: This is in part because you have described KNACK & OPR WRINGER as if they were indefinitely knowledge & intelligent system — so ~~[illegible]~~ I cannot put any constraints on what is going ~~[illegible]~~ on. — Even though I know, of course, that the systems are not so generally intelligent.

environment: electromagnetic pulse (EMP). To evaluate a system, the WRINGER gathers detailed information about an electromechanical system design ranging from the level of major components to the level of individual semiconductors. After the gathered information is checked for completeness and consistency, a worst-case analysis is carried out for each interface circuit in a system, determining whether the EMP environment will induce transients above the operating voltage of the interface circuits. This analysis indicates either that a system is sufficiently refractory of the EMP environment or may not be. In the latter case a more detailed screen analysis, and if necessary an even more precise resistive analysis, is conducted to identify inadequacies in a system's response to the EMP environment. When such an inadequacy is pinpointed, the WRINGER suggests possible fixes, all of which are prechecked for adequate strengthening properties in the interface circuits. <who are these?>

The above general description of a WRINGER and the specific example of the DPR WRINGER indicate that a WRINGERs problem-solving method has to perform two major tasks: gather the requisite information from the designer to describe an incomplete design, and perform a constructive evaluation of the design. The following describes the problem-solving method and the associated knowledge roles for each of these tasks.

<see back of p3

<The paper really floats at this point. I basically have no idea what you are talking about -- i.e., it is too top-down. The terms are a too abstract & I can instantiate them.>

2.1. Information Gathering

A WRINGER first determines which piece of information to gather next. The goal is to reduce the burden placed on the designer in describing a system design. Generally only a fraction of a WRINGER's knowledge is applicable to the evaluation of any particular design. To determine the knowledge relevant to the task at hand, a WRINGER selects the necessary information about a design in a data-driven manner; the decision to gather a particular piece of information is based on previously gathered information. But it is often the case that more than one piece of information can be gathered at any time. The order in which the pieces of information are gathered may be important: the designer might feel more natural providing information in a certain order. For that reason a WRINGER follows the outline of the report it is expected to produce. That skeletal report organizes a WRINGER's knowledge around related topics. In following the outline of the report a WRINGER collects related information in a coherent manner.

Next, a WRINGERs problem-solving method selects and applies a strategy to gather the piece of information determined in the previous step. To take into account the different sources of information available to an evaluator, a WRINGER has several ways to elicit information from the designer or to infer it: For example, it elicits information from the designer by (1) asking a question, (2) interpreting a graphical design description, e.g. a drawing of a system's components, (3) asking the designer to fill in the slots of a table, diagram, or form, or (4) asking the user to choose among the items in a menu. A WRINGER infers additional information based on previously gathered information. For example, it fills in gaps by (5) directly applying specific domain knowledge, (6) computing numeric values, or (7) referring to a database. Often more than one way exists to gather a piece of information. When this is the case, a WRINGER selects a strategy how to gather it. It prefers inference strategies to avoid asking the designer unnecessary questions. Once a strategy is selected, a WRINGER applies the strategy, checks the provided pieces of information for synonyms, and updates the design description. <are these the strategies?

After that, another piece of information to gather next is determined. This iterative process ends when the information required for the evaluation is available.

The above described problem-solving method defines the following roles knowledge can play in the information gathering task:

- Skeletal Report
- Information Identification
- Strategy
- Synonym

{ Any sort of "knowledge about"? I begin to see the edges emerge, but there's also still m

The SKELETAL REPORT role includes knowledge about which chapters, sections and subsections are part of the report a WRINGER is expected to produce, their order, and how to appropriately place report fragments within the report. Dependent on the designer's description of a particular design a WRINGER determines the appropriate chapters, sections, subsections, selects the fragments describing the particular design into the skeleton, and assembles the report. The example below shows a part of the table of contents for the DPR WRINGER.[2]

```
1. Evaluation of the EMP Hardness
   1.1. Summary of the System Description
   1.2. Shielding Requirements for the
        Enclosures
        1.2.1. Diffusion through the Skin of
               the Enclosure
        1.2.2. EMP Leakage through the
               Apertures
...
```

< again, absolutely no constraint on what sort of knowledge. It could be anything I imagine a human could think.>

INFORMATION IDENTIFICATION knowledge is used to identify pieces of information which are relevant to each part of the report. It organizes the required information around related topics.
<another completely general term>

The STRATEGY role describes the different ways to gather a piece of information. Relying on previously elicited information and other pre-defined knowledge, it defines the circumstances in which these techniques can be applied. It also includes instructions about what to expect as a response and what to do with the elicited information. The following example gives a flavor of the kind of questions the DPR WRINGER asks.

```
What enclosures are part of the COMMUNICATIONS
UNIT system? : S-280C, Metal Box

Please, list the apertures of the S-280C
enclosure. : window, cable entry panel
```

The STRATEGY knowledge further includes information about the validity of newly gathered information. This includes finding out whether input provided by the designer is obviously wrong or merely implausible. In the first case, an answer might be outside of a predefined numeric range or it might not be a member of a predefined complete set of possible answers. In the second case an answer might be flagged as questionable because it is not a member of a predefined incomplete set of possible answers. For example: <a little better>

[2]In this and the following examples a WRINGER's and KNACK's prompts and messages appear in boldface, the users responses in underlined boldface. A word in brackets at the end of a WRINGER's or KNACK's prompts is the default response. The user may reply with the default by hitting return: <cr>.

```
Please, list the cables of the COMMUNICATIONS
UNIT system? : audio cable, power cable

I am not familiar with the term AUDIO CABLE.
Some of the following terms are expected
answers:
    SIGNAL CABLE,
    POWER CABLE
Please confirm or revise your answer.
[ AUDIO CABLE ]: signal cable
```

The role SYNONYM is a way to represent knowledge about making a user's answer consistent with the common way of expressing that answer. For example, if a designer's answer to a question contains a synonym for a known expression, the WRINGER replaces it with that known expression. < better >

2.2. Design Evaluation

< this would seem to imply it understands the semantics of all the rich knowledge it has been given >

The collected information is evaluated by a WRINGER for validity, consistency, completeness, and possible design flaws, i.e., a WRINGER checks the information describing a design for violations of constraints imposed by a given environment. If indications of design flaws are found, a WRINGER points them out to the designer. It selects among a number of fixes associated with each constraint violation. A truth maintenance system describes how a fix affects the data item that violated a constraint. If a fix resolves the constraint violation a WRINGER will suggests it to the designer.

Since a WRINGER did not construct the design from scratch, it is not aware of all the implications a proposed change might have. It therefore asks the designer to select one of the suggested fixes. A WRINGER then updates the system description. An applied fix might make additional information necessary and a WRINGER tries to elicit this information from the designer. If the designer cannot or does not want to provide the required information, a WRINGER assumes a worst-case value.

Finally, the description of the system design and the evaluation results are usually documented in some form. A WRINGER presents the gathered information and the results of the evaluation in the form of a report. It uses the skeletal report to determine the structure and the report fragments to determine the content of a report about an actual design. It instantiates the selected report fragments with the information acquired from the designer and with values resulting from its evaluation. A WRINGER then generates the report. In some cases the designer will not agree with the results of the evaluation. For this reason a WRINGER allows the designer to add comments to a report.

The above described problem-solving method defines the following roles knowledge can play in the design evaluation task:

- Constraint
- Fix
- Report Fragment

CONSTRAINT knowledge defines how to uncover contradictory information. For example, the DPR WRINGER makes sure that a cable carrying power is not connected to a cable carrying a signal. CONSTRAINT knowledge further describes how to detect when information provided by the designer is incomplete. For example, the DPR WRINGER checks whether a power source is specified for the system and whether a defined antenna is connected to the rest of the system. Finally, CONSTRAINT knowledge

{I have no idea how complex the constraints or how interlocked they are. You illustrate, but give me no sense of the limits

For all I know it can solve arbitrarily complex non-linear constraint system with thousands of variables, etc)

describes how to identify problem cues that are associated with possible flaws in a design, i.e., how to identify violations of constraints imposed by a given environment. For example:

```
Screen analysis for SIGNAL LINE 2 INTERFACE
CIRCUIT :

Junctions     Vbd  Safe-E.    Threat-E. Eval.
              [V]  [J]        [J]


------------------------------------------
D3  1N752A   200  1.34e-02  8.29e-03  HARD
D4  1N1184   200  3.60e-04  8.29e-03  SOFT
```

The above diagram indicates problems with the SIGNAL LINE 2 INTERFACE CIRCUIT: The circuit contains two diodes in series with the SIGNAL LINE 2. The threat energy produced by the given EMP environment and coupled into the circuit via SIGNAL LINE 2 would damage diode D4.

FIX knowledge suggests how a design could be improved in the case a flaw was found. It also determines worst-case values for the necessary pieces of information the designer could not provide. For example, if the energy coupled into an interface circuit through a cable exceeds an upper limit, the semiconductor devices of the interface circuit will be damaged. The DPR WRINGER will suggest using a terminal protection device to limit that energy to an acceptable level. The following demonstrates this for the above example.

```
The following TPDs will reduce the threat
energy for SIGNAL LINE 2 INTERFACE CIRCUIT
sufficiently:

Type      E.diss    Vover     Vknee     Ton
          [J]       [V]       [V]       [ns]
-------------------------------------------
15KP280  6.00e+10 5.00e+02 3.45e+02 1.00e-03
V420LA10 6.00e+10 2.42e+03 1.20e+03 1.00e-00

Which TPD would you like to use? [ 15KP280 ]:
<cr>
```

A REPORT FRAGMENT describes a small possible piece of an actual report. This includes the text to be printed in a report and the variables containing the information that is specific to whichever system design is the subject of a WRINGER report. It also incorporates the gathered information into the report. The report part used as the sample report in section 3 is also an example of a report part produced by the DPR WRINGER.

3. Acquiring Knowledge

KNACK is a knowledge acquisition tool that can be used by domain experts to create WRINGERs, expert systems that assist with the evaluation of different classes of designs. An important goal in the development of KNACK is, that is acquires knowledge from domain experts without presupposing knowledge engineering skills on their part. To reach that goal, KNACK's approach for knowledge acquisition combines and uses existing AI techniques to derive a general description how to evaluate designs from a specific sample description. This is a process of abstraction (e.g. the specific sample description is variabilized) and completion (e.g. signs of incompleteness lead to the elicitation of additional knowledge).

General knowledge about evaluating designs is incorporated into KNACK. In an initial questioning session with the expert, KNACK uses that knowledge to conduct an interview with the expert. The interview results in a preliminary model of a particular evaluation domain. During knowledge acquisition KNACK refines the preliminary domain model into a detailed structural and functional model of the domain. The domain model describes the concepts, their interdependencies, and the vocabulary the expert uses in performing an evaluation task. KNACK also requires a sample report as an initial input. The sample report is a document that exemplifies the description and the evaluation of a particular design.

Once the sample report is typed in and an initial domain model is defined, KNACK develops expertise in evaluating designs by integrating the specific sample report with the domain model in successive interactions with the expert. This integration process generalizes the sample report, making it applicable to different systems. To demonstrate its understanding of the sample report and to predict and exemplify the performance an expert can expect from the WRINGER he is creating, KNACK instantiates the generalized report with known concept representatives taken from the domain model. It displays several differently instantiated examples for each generalized report fragment. The expert edits any examples that make implausible statements. KNACK uses this feedback as additional knowledge to correct its generalizations and refine the domain model.

Once the expert accepts KNACK's understanding of the sample report, KNACK elicits knowledge about how to customize the generalized sample report for a particular application. The expert defines strategies that a KNACK generated expert system, a WRINGER, will use to acquire values instantiating the concepts in the generalized fragments. Experts define strategies in the same way that report fragments are defined: by typing in samples. Strategies can be questions, formulas, inferences, and other forms. KNACK generalizes the strategies and displays some example instantiations of them for review and correction by the expert.

KNACK's knowledge acquisition approach results in a knowledge base the generated WRINGER expert system can use to evaluate a range of designs and to present the results in the form of a report. However, the sample report covers only one simple design and almost certainly lacks some important concepts needed for the evaluation of a broader range of designs. For this reason, KNACK searches the knowledge base for report fragments or strategies that indicate gaps or conflicts with its domain model. If a possible flaw is found, KNACK asks the expert to correct the report, the strategies, or the domain model.

The following detailed description of KNACK's knowledge acquisition approach is organized around an example of an actual KNACK case: the creation of the DPR WRINGER. It leads through the process of typing in a small part of a sample report, acquiring a partial domain model, generalizing the part of the sample report, and defining strategies. The analysis of the acquired knowledge is demonstrated in section 4. In the interest of brevity, the excerpts used as examples are only a small fraction of the full KNACK case.

KNACK starts out with displaying the top level menu.

```
model                  : acquire domain model
report                 : acquire report
generalize-report      : generalize report
strategy               : acquire strategies
generalize-strategy    : generalize strategies
analyze                : analyze knowledge base
exit                   : exit KNACK
```

KNACK's report-driven approach to acquire knowledge determines the expert's choices on the top level: define or update the domain model, sample report, or strategies; generalize the sample report or sample strategies; analyze the knowledge base for incompleteness or inconsistency. Once the sample report is typed in and an initial domain model is defined, the expert can choose any of the above functions. Our sample interaction starts with typing in the sample report.

3.1. Acquiring the Sample Report

KNACK requires a sample report as an initial input. The sample report is a document describing how the expert evaluates a particular design. It exemplifies what the expert intends the WRINGER to produce. It may be written specially for this purpose by a domain expert or group of experts, or selected from existing reports. It contains a description of the design and the given environment, a detailed evaluation of the design with regard to the environment, and suggestions to improve the design in case design flaws are found. The sample report is a familiar and convenient medium for the expert to express his knowledge.

The selection of the REPORT option in the top level menu leads to the report menu shown below.

```
next     : display next fragment
previous : display previous fragment
edit     : edit current fragment
insert   : insert fragments
delete   : delete current fragment
quit     : quit sample report editor
```

It determines the top level features of a simple text editor that can be used to define, update, or leaf through the sample report. The INSERT function allows to input the sample report. The sample report is typed in to a file by any person familiar with text editors.

```
1. Evaluation of the EMP Hardness

1.1. Summary of the System Description

The system Communications Unit is designed to
resist to EMP threat.  It consists of a
Computer, a Modem, a Radio, and a Motor
Generator.  Power is supplied from the Motor
Generator to the Computer, Modem, and Radio by
the Power Cable.

The Computer, Modem, and Radio are protected
by a S-280C enclosure.  The Motor Generator is
protected by a Metal Box enclosure.

The S-280C enclosure has the following
apertures: Window and Cable Entry Panel.  The
Metal Box enclosure has the following
apertures: Cable Entry Panel.

1.2. Shielding Requirements for the S-280C
     Enclosure

1.2.1. Diffusion through the Skin of the
       Enclosure
```

```
The S-280C enclosure is made of aluminum and
is 30 mils thick.  Aluminum has a
relative-conductivity of 0.15 mhos/m.  A plate
of aluminum must be at least 20 mils thick to
reduce the diffusion factor to an negligible
level.  Therefore, the diffusion factor can be
neglected.
```

KNACK divides the report into fragments corresponding to paragraphs. In the above example, this results in 8 report fragments.

KNACK also requires a model of the domain as an initial input. Thus, our example continues with the definition of the domain model.

3.2. Acquiring the Domain Model

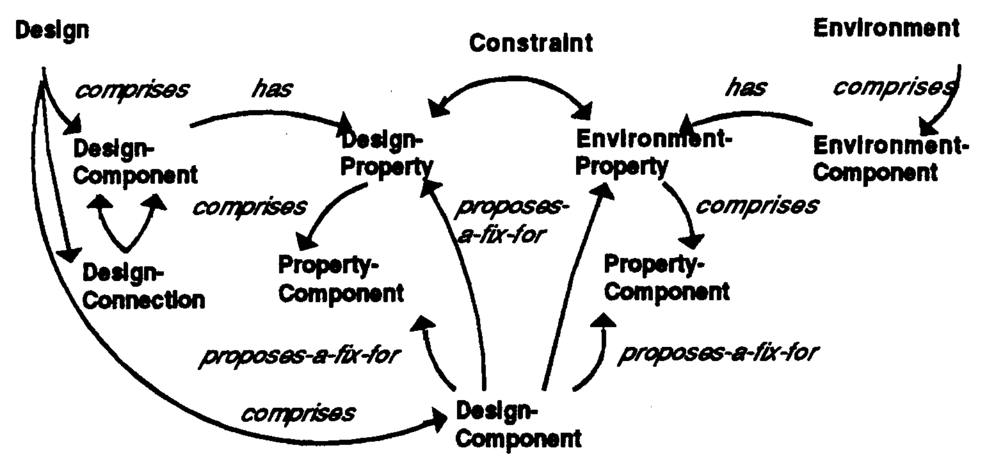
The sample report describes a particular system design in the terms familiar to the expert. To generalize the sample report, making it applicable to other designs, KNACK needs a model of the particular evaluation domain. The domain model contains a detailed structural and functional description of the evaluation task at hand. The structural part of the domain model describes a taxonomy of the concepts, vocabulary, and terms experts use and the interdependencies between concepts. This includes the relevant parameters of a design and the environment, the constraints, and possible fixes for violated constraints. The functional part of the domain model describes procedures how to determine, compare and propagate relevant parameters. Thus, the model customizes KNACK for a particular evaluation domain.

In addition to generaliz~~e~~ing the sample report, KNACK uses the concepts and terms described in the domain model to acquire knowledge in a format familiar to the expert. The model further represents a preview of the knowledge base the expert wants to create. KNACK uses it to develop expectations about the knowledge the expert might provide. Based on these expectations KNACK checks the expert's input, generates knowledge pieces, and analyzes the resulting knowledge base. The expert then refines KNACK's expectations (as described in sections 3.4, 3.5, and 4) and, thus, refines the domain model.

At the beginning of knowledge acquisition, KNACK acquires a preliminary domain model. To acquire the preliminary model, it conducts an interview with the expert. This is an interactive process which takes a few hours of the expert's time. The interview is driven by KNACK's general understanding of the evaluation task. That understanding describes the knowledge common to a range of evaluation domains on a high level of abstraction. KNACK views evaluation as partly analytic (i.e., determine whether a system will function in a given environment) and partly constructive (i.e., improve a system design so that it will function in a given environment). The example below gives a flavor of the knowledge applicable to a range of evaluation tasks:

< I would have to believe from this the KNACK is a general intelligent agent who understood about evaluation what an engineer does. >

One of the problems is that paper evinces no awareness of the problem that is plaguing the reader — that there is an issue of conveying the limitation of the mechanism.



The general understanding is buil~~d~~t into KNACK. It describes a design that needs to be refined to take into account aspects of an unfamiliar environment. The nodes are concepts and the links between the nodes encode structural knowledge. For example, a design comprises a set of design components interrelated by design connections. Design components have design properties. An environment comprises environment components. They are further described by environment properties. Design and environment properties define the evaluation criteria and are compared to some other properties. This defines the constraints a design has to satisfy. Properties can comprise property components. Design components can propose a fix for properties or property components. A fix changes a design or suggests a design extension and, thus, modifies design or environment properties.

KNACK uses the above described general understanding in the form of generic questions to acquire the vocabulary experts are familiar with. The expert's answers customize KNACK's abstract knowledge into a preliminary model of a particular domain. The preliminary model describes concepts, concept characteristics, concept representatives, and constraints. The following sample interaction defines a part of the preliminary model needed to generalize the above sample report.

Is this limited to synonyms? This is a very limited (though important)

```
How would you like to refer to SYSTEM
components? subsystem, enclosure

How would you like to refer to SYSTEM
components that interrelate SYSTEM components?
cable

How would you like to refer to the environment
in which a SYSTEM must function? nuclear
environment
```

The answers to the above questions define new concepts for the domain model. They denote a design, an environment, design and environment components and connections.

Other questions define concept properties as the data items to be compared by the evaluation criteria.

```
How would you like to refer to the SUBSYSTEM
properties that define the data items to be
compared by the evaluation criteria? safe
energy
```

```
How would you like to refer to the COUPLING
properties that define the data items to be
compared by the evaluation criteria? threat
energy
```

Once the constrained data items are known, KNACK asks the expert to determine the constraints. Constraints define the evaluation criteria, i.e., the relationships between the data items to be compared. The constraints are defined by keywords like "equal", "less equal", "subset", etc.

< how "etc". So — "less than the exponentially weighted integral of" ok? >

```
What is the relationship between the SAFE
ENERGY property and the THREAT ENERGY
property? less equal
```

Further questions determine the concepts that represent fixes in case constraints are violated.

```
What design components represent a fix for the
THREAT ENERGY property? enclosure
```

Concepts are described further by their characteristics.

```
What are the characteristics of the Enclosure?
material, thickness
```

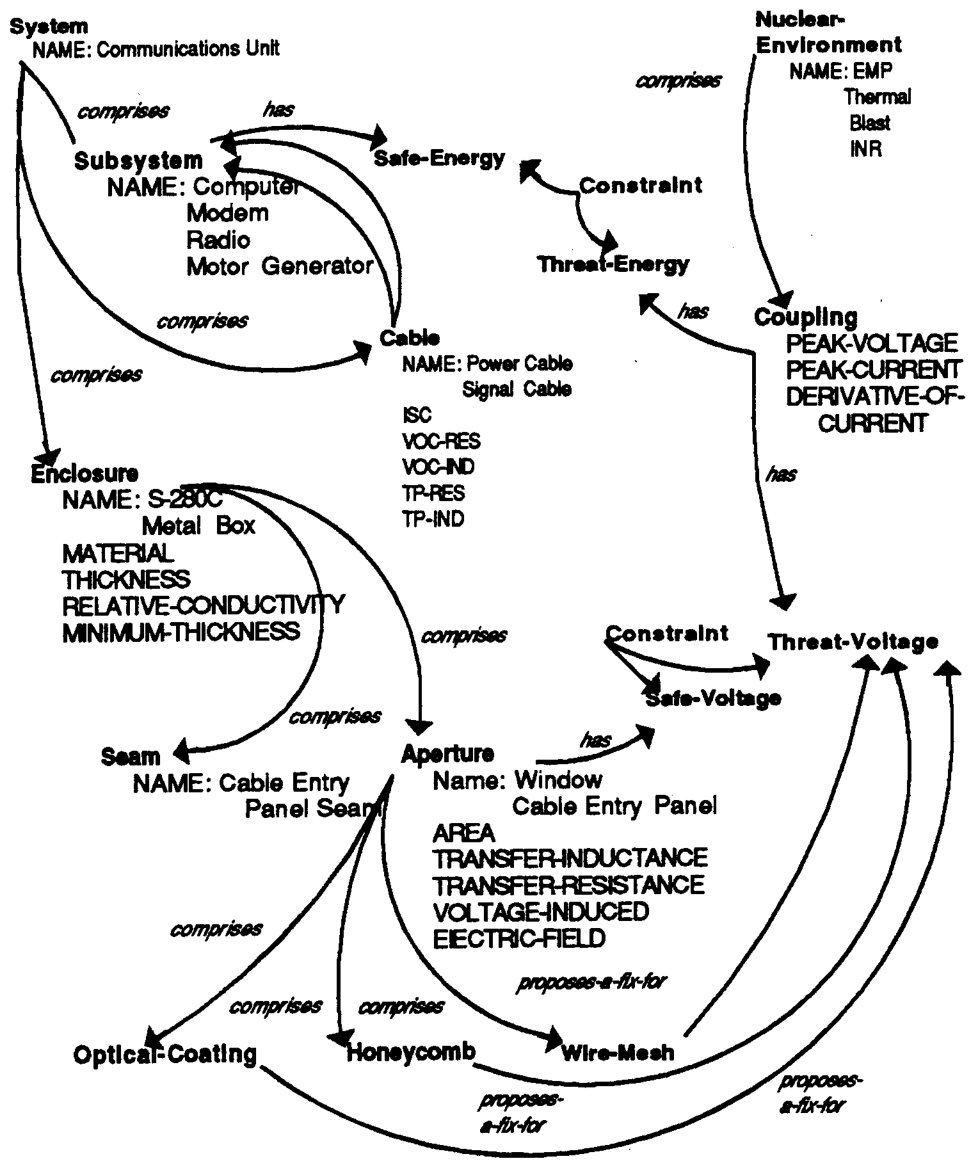
< is this wide open? introducing terms KNACK knows nothing about? >

Some questions determine the concept representatives used in the sample report. These are important keywords KNACK uses in its generalization process.

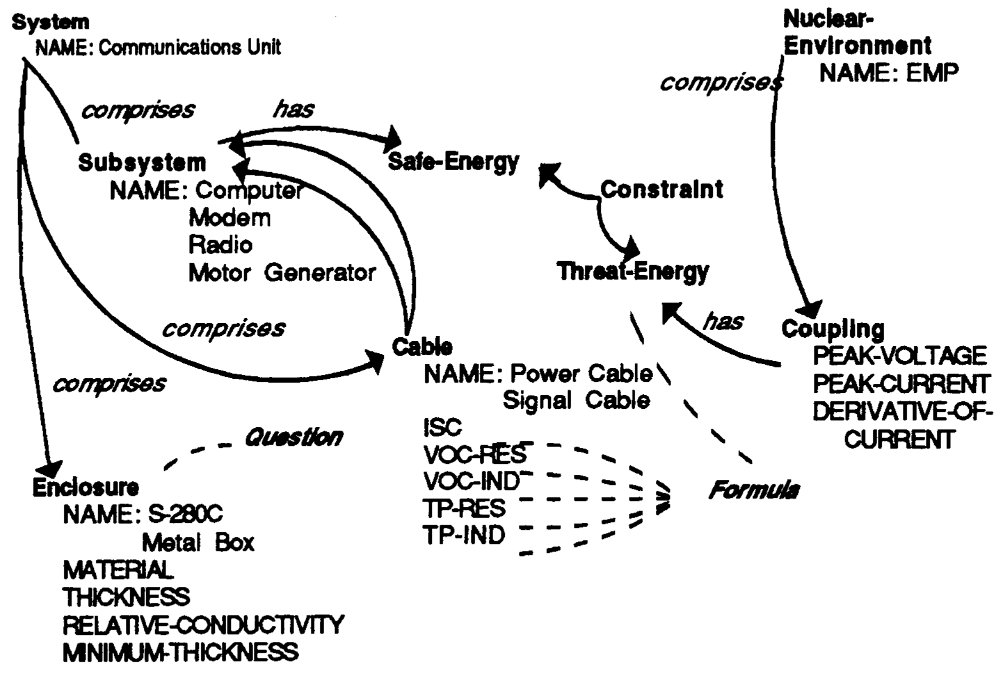
```
Please, list some examples for the SUBSYSTEM:
computer, modem, radio, motor generator

Please, list some examples for the ENCLOSURE:
S-280C, metal box

Please, list some examples for the MATERIAL
characteristic of the ENCLOSURE: aluminum

Please, list some examples for the THICKNESS
characteristic of the ENCLOSURE: number
```

< I infer that one of the things KNACK *really* knows is that "components" have "examples" >

The expert's responses to all of the above questions are added to KNACK's general understanding of evaluation resulting in a preliminary domain model. The following is an example of a preliminary model, parts of which are derived from the above interaction.



The preliminary domain model is not sufficient for a successful generalization process. It represents a structural description of the task i.e., the vocabulary and the terms experts use, but does not contain any functional knowledge, i.e., how experts obtain and propagate design and environment parameters. The following example demonstrates a portion of the domain model representing functional knowledge. The nodes describe procedures to obtain design and environmental parameters. The links (indicated by dotted lines) define how values are propagated through the network. Section 3.5 describes the process of deriving that functional knowledge.



The above example states that a question strategy can be used to determine the kinds of ENCLOSUREs comprised in a SYSTEM. It describes further that a formula strategy uses values of the ISC, VOC-RES, VOC-IND, TP-RES, and TP-IND characteristics of CABLE to determine the value for the THREAT-ENERGY.

It is likely that the initial structural model is incomplete and not detailed enough. During knowledge acquisition KNACK augments the initial domain model to include the functional aspects of the evaluation domain and to obtain a more detailed structural model. It gradually specializes the domain model to represent the expert's understanding of how a particular environment interacts with different design components. This process is described in section 3.4.

Once the sample report is typed in and an initial domain model is defined, KNACK interacts with a domain expert to generalize the sample report on a fragment by fragment basis.

3.3. Generalizing the Sample Report

KNACK develops expertise in evaluating designs by integrating the specific sample report with the domain model in successive interactions with the expert. This integration process generalizes the sample report, making it applicable to a broader range of designs. Deriving the generalized report involves extracting the report's basic structure and integrating the domain model with the report fragments (i.e. fragments are parsed to detect text strings that match the entries in the domain model). The technique employs simple heuristics to infer the concepts each fragment mentions. The heuristics are based on keywords, representatives for concepts in the fragment, and knowledge of relations between candidate concepts.

In the first aspect of this process KNACK looks for keywords (e.g., chapter, section, subsection, heading, itemize, enumerate, bold, underline), instances of keywords (e.g., 2. for chapter, 2.3.2. for subsection, (1) for enumerate), and the form of the input (only a few words in a line separated from the remaining text by blank lines). From this analysis KNACK generates a skeletal report defining the form of the sample report. It includes the outline and special formats (e.g., table of contents, itemizations, enumerations, filled or unfilled environments) encoded as commands for a document formatting system.

In the second aspect of the generalization process KNACK converts fixed report text into generalizations representing the concepts detected in the fragment. Cues to locate and identify concepts in a report fragment are numbers representing the value of quantitative parameters and non-numeric symbols denoting tokens of known concepts in the domain model.

The heuristics provide sufficient analytical power to acquire knowledge without turning to a sophisticated natural language interface. There are limitations though. The heuristics mistakenly identify some concepts and miss others. The errors are dealt with when the expert critiques instantiations of the generalized fragments as described later on.

The generalization process results in a collection of generalized report fragments more broadly applicable than the sample report. A generalized report fragment describes a small possible piece of an actual report. It includes fixed text strings to be printed exactly as formulated by the expert, concepts to be instantiated by the WRINGER, knowledge about incorporating the gathered concept representatives into the report, and keywords specifying the type and form of the report fragment (e.g., simple paragraph, figure, table, and title). Generalizations are internal constructs for KNACK's use. Consonant with the research goal of reducing the knowledge engineering skills needed for knowledge acquisition, the expert sees only instantiated generalizations. However, the following examples of generalizations are included to give an impression of the kinds of heuristics KNACK uses to integrate the domain model with the sample report.

To give a flavor of the kinds of heuristics KNACK uses, generalizations of some of the fragments in the above sample report are shown below. The angle brackets enclose concepts detected in a fragment. Asterisks enclose commands denoting the report structure.

```
*CHAPTER* Evaluation of the <ENVIRONMENT.NAME>
Hardness
```

Numbers at the beginning of a line followed by a dot indicate chapter, section, or subsection headings. For example, "1." is assumed to be a chapter heading. Representatives of known concepts can be generalized by replacing them with a variable representing that concept. For example, EMP is inferred to be a NAME of an ENVIRONMENT due to a unique match with the domain model.

```
The system <SYSTEM.NAME> is designed to resist
to <ENVIRONMENT.NAME> threat.  It consists of
*LOOPOVER* <SUBSYSTEM.NAME> a <SUBSYSTEM.NAME>
,*ENDLOOP*.  Power is supplied from the
<SUBSYSTEM.NAME> to the *LOOPOVER*
<SUBSYSTEM.NAME> <SUBSYSTEM.NAME>, *ENDLOOP*
by the <CABLE.NAME>.
```

A list of representatives for the same concept is replaced with a variable representing that concept and a surrounding LOOP structure. In a WRINGER report, a fragment containing a LOOP structure will be printed once, whereas the text within the LOOP structure will be repeated for each instantiation of the variable. For example, "a Computer, a Modem, a Radio, and a Motor Generator" is assumed to be a list of NAMEs of SUBSYSTEMs.

```
The *LOOPOVER* <SUBSYSTEM.NAME>
<SUBSYSTEM.NAME>, *ENDLOOP* are protected by
a <ENCLOSURE.NAME> enclosure.
```

If the generalizations of parts of the report fragments are the same, these parts can both be represented by the same generalization. For example, the generalization of the two sentences "The Computer, Modem, and Radio are protected by a S-280C enclosure. The Motor Generator is protected by a Metal Box enclosure." results in the same generalized report fragment shown above.

```
The <ENCLOSURE.NAME> enclosure is made of
<ENCLOSURE.MATERIAL> and is
<ENCLOSURE.MINIMUM-THICKNESS> mils  thick.
<ENCLOSURE.MATERIAL> has a relative
conductivity of
<ENCLOSURE.RELATIVE-CONDUCTIVITY> mhos/m.  A
plate of <ENCLOSURE.MATERIAL> must be at least
<ENCLOSURE.THICKNESS> mils thick to reduce the
diffusion factor to an negligible level.
Therefore, the diffusion factor can be
neglected.
```

A number is assumed to be a representative of some numerical characteristic of some concept. If the text adjacent to a number refers to a known concept and one of its characteristics, the number is replaced with the corresponding variable. For example, "0.15" is assumed to be the RELATIVE-CONDUCTIVITY of an ENCLOSURE because ALUMINUM is known to be an example for an ENCLOSURE MATERIAL and the term "relative conductivity" was encountered in the text of the fragment.

When helpful clues are not present in adjacent text, KNACK simply guesses the concept from the ambiguous set of matches. Its guesses are based on the concepts recognized in the fragment. These guesses can be wrong and KNACK corrects them when the expert critiques instantiations of the generalized fragments as described later on. The above fragment contains the guesses <ENCLOSURE.MINIMUM-THICKNESS> and <ENCLOSURE.THICKNESS>.

3.4. Demonstrating Understanding of the Sample Report

KNACK predicts and exemplifies the performance an expert can expect from the WRINGER he is working to create. It instantiates the concepts of the generalized fragments with known concept representatives taken from the domain model and displays several differently instantiated examples of each generalized report fragment. The expert edits any examples that make implausible statements about the domain. KNACK treats such events as incorrect use of the knowledge base and interprets the corrections as new knowledge to update the generalization and improve the domain model. For example if the expert indicates that values from the domain model combine too loosely, KNACK adds a constraint to the model, restricting possible combinations. A correction also can imply that KNACK's guess about the identity of a concept is wrong, leading to its retraction and the introduction of a new, initially less probable guess. Applying the new knowledge, the generalization is instantiated again and display of several examples gives the expert immediate feedback on the effects of the knowledge base modification.

KNACK extends the domain model whenever the editing adds variability between examples that it cannot parse. Extensions can be new concepts, new characteristics for known concepts, and restrictions on existing relations between representatives of two concepts. The model serves as a collection of examples suggesting guesses for KNACK as to the form of the extension. The following examples illustrate the editing process with some of the generalized report fragments shown above.[3]

```
1. Evaluation of the EMP Hardness

2. Evaluation of the Thermal Hardness

Corrections? [ NONE ]: delete example 2
```

KNACK assumes that this fragment represents a chapter heading and that there will be a chapter about the different environments defined in the domain model: EMP, THERMAL, BLAST, and INR. Therefore, it displays examples of possible chapter headings. The expert deletes the second example. KNACK guesses that only the value EMP is relevant for the fragment. It further constrains the remaining fragments of the chapter to the EMP environment, assuming that the topic will not change within a chapter.

```
The system Communications Unit is designed to
resist to EMP threat.  It consists of a
Computer, a Modem, a Radio, and a Motor
Generator.  Power is supplied from the Motor
Generator to the Computer, Modem, Radio, and
Motor Generator by the Power Cable.

The system Communications Unit is designed to
resist to EMP threat.  It consists of a
Computer, a Modem, a Radio, and a Motor
Generator.  Power is supplied from the
Computer to the Computer, Modem, Radio, and
Motor Generator by the Signal Cable.

Corrections? [ NONE ]: delete the second
occurrence of TO in example 1, delete the 2nd
occurrence of MOTOR GENERATOR in example 1,
change the 2nd occurrence of COMPUTER in
example 2 to MOTOR GENERATOR, change SIGNAL
CABLE in example 2 to POWER CABLE
```

Since KNACK generalized the fragment to contain a list of NAMEs of SUBSYSTEMs, it includes all representatives of SUBSYSTEMs it knows into the example assuming that the fragment is valid for all possible SUBSYSTEMs. The expert first makes a minor change to the fixed text of the fragment. This example demonstrates further that KNACK's domain model is inadequate. The expert's corrections are now used to refine the model: KNACK integrated the concepts SUBSYSTEM and CABLE with the report fragment. It knows, that some CABLE interrelates some SUBSYSTEM with some SUBSYSTEM, and that no other relation interrelates SUBSYSTEM with another concept integrated with the report fragment. KNACK now adds the restriction to the model that a POWER CABLE interrelates a MOTOR GENERATOR SUBSYSTEM with SUBSYSTEMs different from MOTOR GENERATOR. In this case, the process of extending the domain model is internal to KNACK. The corrections provide KNACK with all the information necessary to extend the model.

KNACK cannot always decide unambiguously which relation to restrict because more than one relation

[3]The expert uses the mouse and provided menus in order to change displayed examples. It is beyond the scope of this paper to present this part of the interaction exactly as it proceeds on the terminal screen. Thus, in the following examples the expert's actions are described in short, underlined sentences.

interrelate concepts detected in the report fragment. In this case KNACK guesses a relation to restrict. It assumes that its guess is right, until a correction of an instantiation later in the interaction indicates the opposite. KNACK then revises its earlier decision and restricts another relation.

```
1.2. Shielding Requirements for the S-280C
Enclosure

1.3. Shielding Requirements for the Metal Box
Enclosure

Corrections? [ NONE ]: <cr>
```

KNACK displays two examples corresponding to the known representatives of ENCLOSUREs. The expert agrees with both. This section heading contains a variable text component. In a WRINGER report the section begun by the above example will be repeated for all ENCLOSUREs contained in an actual system. The report fragments within each of the sections will be specific to the enclosure described by a section.

Once the sample report is generalized, KNACK suggests to define strategies which a WRINGER will use to customize the generalized report fragments for a particular application.

3.5. Defining, Generalizing, and Correcting Strategies

Concepts in the generalized fragments must be instantiated with values describing a particular system design when a WRINGER evaluates a design and writes its report. KNACK asks the expert to define strategies for a WRINGER to acquire or produce the instantiation values. Experts define strategies in the same way that report fragments are defined, by typing in samples. Each strategy describes a way to determine a representative of a concept and includes instructions about valid possible values. Relying on previously elicited information and other prior knowledge, KNACK defines the circumstances in which these methods can be applied. The strategies are also used to refine the domain model: they describe the procedures to obtain, propagate and compare design and environment parameters.

KNACK asks the expert to define at least one strategy for each concept in the report fragments. A strategy can be interactive, i.e., acquire concept representatives by asking questions, interpreting a graphical design description, asking the designer to fill in the slots of a table, diagram, or form, or asking the user to choose from the items in a menu. Other strategies are autonomous, i.e., infer concept representatives by directly applying specific domain knowledge, computing numeric values using formulas, or referring to a database.

```
Which strategy can be used to determine the
ENCLOSUREs of a SYSTEM?

[constant, question, inference, table, menu,
graphics, formula, database, postpone, quit]
[ QUESTION ]: <cr>

question text....: Please, list the enclosures

possible answers.. [ INCOMPLETE-SET, S-280C,
                   METAL BOX ] <cr>
default answer.... [ S-280C ]: unknown
status............ [ NOT-MANDATORY ]: <cr>
```

The above example demonstrates how the expert defines the knowledge needed for a question strategy to determine the ENCLOSUREs of a SYSTEM. He defines the question "Please, list the enclosures". KNACK suggests defaults for the expert's input. These are taken from report fragments or the domain model. For example, KNACK knows that S-280C and METAL BOX are examples for an ENCLOSURE and suggests these as possible answers. The expert agrees that S-280C or METAL BOX or both are possible answers. He further defines that an answer to the question is not mandatory. Also, it is not meaningful to define a default answer.

KNACK develops expertise in evaluating designs by parsing the text of the question in an attempt to generalize it. It integrates the specific sample strategy with the domain model, thus making the sample strategy applicable to acquire instantiation values for a broader range of concepts. On the other hand, a strategy must be discriminating enough to result in the instantiation of the right concept. KNACK uses heuristics to make the text of a question strategy more specific. For example, since the domain model states that a SYSTEM comprises ENCLOSUREs, KNACK generalizes the text of the above question to:

```
Please, list the enclosures of the
<SYSTEM.NAME> system
```

The specialization of the question text is guessed by KNACK and can be wrong or unnecessary. Thus, KNACK instantiates the concepts integrated with the question text with known representatives and displays these examples for confirmation or correction by the expert.

```
Please, list the enclosures of the
Communications Unit system?

Corrections? [ NONE ]: <cr>
```

Continuing the above example, KNACK adds to the domain model that ENCLOSUREs can be determined using a question strategy. The updated domain model was shown in section 3.2.

The interaction continues with an example of an autonomous formula strategy to determine the THREAT-ENERGY.

```
Which strategy can be used to determine the
THREAT-ENERGY of a COUPLING?

[constant, question, inference, table, menu,
graphics, formula, database, postpone, quit]
[ QUESTION ]: formula

THREAT-ENERGY = Isc * ( ( Voc-res * Tp-res ) +
( Voc-ind * Tp-ind ) )
```

KNACK parses the formula to generalize it. Since all the terms in the formula are characteristics of the CABLE concept, KNACK variablizes the formula to:

```
<CABLE.ISC> * ( ( <CABLE.VOC-RES> *
<CABLE.TP-RES> ) + ( <CABLE.VOC-IND> *
<CABLE.TP-IND> ) )
```

The variables of the formula are guessed by KNACK and can be wrong. To confirm its guesses KNACK displays instantiated generalizations.

```
THREAT-ENERGY = Isc * ( ( Voc-res * Tp-res ) +
( Voc-ind * Tp-ind ) )
with Isc     = Isc of CABLE
     Voc-res = Voc-res of CABLE
     Tp-res  = Tp-res of CABLE
     Voc-ind = Voc-ind of CABLE
     Tp-ind  = Tp-ind of CABLE

Corrections? [ NONE ]: <cr>
```

Again, KNACK updates its domain model with the knowledge on how to obtain the THREAT ENERGY parameter using the defined formula strategy.

A final example demonstrates an autonomous inference strategy to determine the enclosures comprises in a system. This strategy describes a procedure to extend an existing design in case the THREAT ENERGY is to high.

```
Which strategy can be used to determine the
ENCLOSUREs of a SYSTEM?

[constant, question, inference, table, menu,
graphics, formula, database, postpone, quit]
[ QUESTION ]: <inference>

if   the constraint THREAT ENERGY LESS EQUAL
         SAFE ENERGY is violated, and
     a COMMUNICATIONS UNIT SYSTEM exists, and
     a COMPUTER SUBSYSTEM exists, and
     the SYSTEM COMPRISES the SUBSYSTEM, and
     the SUBSYSTEM HAS the SAFE ENERGY, and
     no S-280C ENCLOSURE exists,
then add a S-280C ENCLOSURE to the design

if   the constraint THREAT ENERGY LESS EQUAL
         SAFE ENERGY is violated, and
     a COMMUNICATIONS UNIT SYSTEM exists, and
     a MOTOR GENERATOR SUBSYSTEM exists, and
     the SYSTEM COMPRISES the SUBSYSTEM, and
     the SUBSYSTEM HAS the SAFE ENERGY, and
     no S-280C ENCLOSURE exists,
then add a METAL BOX ENCLOSURE to the design

Corrections? [ NONE ]: <cr>
```

KNACK uses the domain model to suggest how a piece of information can be inferred given some previously gathered information. Again, KNACK instantiates the rules with specific examples taken from the domain model, displays several differently instantiated examples, and uses the expert's corrections to refine the domain model.

The definition, generalization, and correction of strategies complete the initial interaction between KNACK an the domain expert. This results in a knowledge base the generated WRINGER expert system can use to evaluate a range of system designs.

4. Analyzing the Knowledge Base

The knowledge KNACK acquires during its interaction with an expert, or group of experts, is transformed into an internal representation and stored in a knowledge base. KNACK's knowledge acquisition approach, described in the preceding sections, generalizes a specific sample report. This results in a knowledge base which the generated WRINGER expert system can use to evaluate a range of system designs. However, the sample report covers only one simple system and inevitably lacks concepts necessary to evaluate a broad range of systems.

For this reason, KNACK searches the knowledge base for report fragments or strategies that indicate gaps or conflicts with its domain model. This review of the knowledge base is most relevant at the end of the acquisition process, because an apparent gap found during the process might be filled in later in the process. When a conflict was detected or an indication of a gap was found, KNACK asks the expert to correct either the fragment, the strategy, or the domain model. In cases where the domain model is changed, KNACK reviews all fragments or strategies that use the changed concept or relation to propagate the change through the knowledge base automatically, making guesses when ambiguities arise. On the other hand, when the expert adds or changes report fragments or strategies, KNACK processes them through the integration of the domain model, display of examples, strategy definition, and checking.

Some of the heuristics KNACK uses to identify incompleteness and inconsistency in its knowledge base are:

- Each concept characteristic in the domain model must have a strategy associated with it to instantiate the concept.
- Each concept or concept representative should be mentioned in the sample report.
- The expert might have forgotten to define concepts, concept characteristics, or concept representatives.
- Constraints must exist to define the relationship between design and environment properties.
- Each constrained property must have a fix associated with it.

To analyze the knowledge base, it must be explicit how different knowledge pieces interact during problem-solving. This is achieved by organizing the knowledge base according to the different roles knowledge plays in the design evaluation task. The domain model introduced in the previous section is the kernel of the knowledge base. It is implemented as a semantic network. The nodes describe concepts used by domain experts to describe, evaluate, and enhance designs and their environments. Concepts are further described by characteristics. Each characteristic represents a variable. The variables must be acquired or infered by the generated expert system. The interdependencies between concepts and variables are defined by the links in the domain model. The links encode structural and functional knowledge. Structural links include COMPRISES, HAS, CONSTRAINS, and PROPOSES-A-FIX-FOR relationships. They define a taxonomy of concepts. Functional links relate strategies to variables. They define which variables are input to a strategy and which variable contains the result of a strategy.

Fix, constraint, and strategy nodes are further described by FIX, CONSTRAINT, and STRATEGY knowledge, respectively. STRATEGY knowledge determines how to obtain, propagate, and compare parameters, CONSTRAINT knowledge describes the relationship between two parameters, and FIX knowledge suggests design components that might resolve a constraint violation.

The domain model is also the kernel of a second knowledge structure that represents the document a WRINGER will produce. That knowledge is defined by the REPORT FRAGMENT, INFORMATION

IDENTIFICATION, and SKELETAL REPORT roles. REPORT FRAGMENT knowledge represents a possible paragraph of the actual report. The variables in the domain model are linked to each REPORT FRAGMENT that uses them. The order of the fragments and the structure of the report (chapter, section, subsection, etc.) are defined by SKELETAL REPORT knowledge. INFORMATION IDENTIFICATION knowledge identifies the variables relevant for different report parts (chapter, section, subsection).

Finally, each concept, concept characteristic, and concept representative can be refered to via different terms. The SYNONYM role organizes the knowledge to convert those synonyms into a basic expression.

The above heuristics exploit that explicit organization of the knowledge base to look for gaps and conflicts. The remaining part of this section explains this in more detail.

A WRINGER must have available at least one strategy to instantiate each concept characteristic in the domain model. KNACK looks for concept characteristics in the generalized report fragments and strategies ~~which~~ that do not have a corresponding strategy.

```
The characteristic MATERIAL of the concept
ENCLOSURE was mentioned in the sample report.
 No strategy exists to acquire that
information.  Do you want to define one now?
[ YES ]: no
```

The expert can define the missing strategy using the process described in section 3.5. In the above example, the expert does not want to deal with the problem right now. KNACK allows the expert to go on with the interview, but will remind the expert of the insufficiency the next time the ANALYZE function is selected.

A flaw in the knowledge base is indicated if a concept or a representative for a concept was introduced into the model but never used. KNACK reminds the expert of that.

```
The representatives THERMAL, BLAST, and INR
for the concept NUCLEAR ENVIRONMENT are known
but never used in any report fragment.  Do you
want to add a fragment? [ YES ]: no
```

Again, the expert postpones work on the potential problem. The answer YES will activate the sample report editor, allowing the expert to add additional fragments or change existing ones. KNACK will generalize the changed or additional fragment and display instantiations for confirmation by the expert as described earlier.

The knowledge base might be incomplete or inconsistent because the expert forgot to mention concepts, characteristics, or representative values. For each concept and characteristic figuring in relations with several others and for the representative values of each concept and characteristic, KNACK asks for possible extensions to that set. For example:

```
A system comprises the following concepts:
SUBSYSTEM, ENCLOSURE.  Do you want to
consider any other system component comprised
in a SYSTEM? [ NO ]: antenna
```

This introduces a new concept: ANTENNA. KNACK integrates new concepts into the model using the process described in section 3.2.

or seems to be.
Here is an interesting part of KNACK's underlying model: Variablization is done over classes & the language that applies to one member of a class, can apply to them all, without modification. Hence the fixed text can be preserved & reused

```
Please, list some examples for the ANTENNA:
whip antenna, dish antenna

What are the characteristics of the ANTENNA?
length, diameter, min operating frequency,
max operating frequency

How would you like to refer to the ANTENNA
properties that define the data items to be
compared by the evaluation criteria? unknown
```

KNACK then examines the generalized sample report to find fragments mentioning the ANTENNA concept. As the domain model previously did not include knowledge about ANTENNAs, any occurrences in the sample report fragments were treated as fixed text in the generalizations. KNACK now variabilizes the new concept in those fragments and displays instantiated examples. The example of the sample report does not mention the concept antenna. If there are no fragments mentioning the new concept, KNACK looks for related concepts in the domain model, i.e., for the concepts figuring in the same relations than the new concept. It then integrates the new concept with fragments dealing with the related concepts and displays instantiations for confirmation by the expert. Using the domain model from the previous examples, KNACK finds that a SYSTEM also comprises SUBSYSTEMs and ENCLOSUREs. It integrates ANTENNA with the first fragment mentioning the SUBSYSTEM concept and displays an instantiation for review by the expert:

```
The system Communications Unit is designed to
resist EMP threat.  It consists of a Computer,
a Modem, a Radio, a Motor Generator, a Wip
Antenna, and a Dish Antenna.  Power is
supplied from the Motor Generator to the
Computer, Modem, Radio, Wip Antenna, and Dish
Antenna by the Power Cable.

Corrections? [ NONE ]: delete the 2nd
occurrence of WIP ANTENNA, delete the 2nd
occurrence of DISH ANTENNA, insert  "Signals
are received by the Wip Antenna and
transmitted to the  Radio via the Signal
Cable." after the example
```

KNACK adds the restriction to the model that a POWER CABLE does not interrelate a MOTOR GENERATOR SUBSYSTEM with ANTENNAs. It then integrates the domain model with the newly defined fragment and displays instantiations for confirmation by the expert. KNACK continues to integrate the ANTENNA concept with fragments dealing with SUBSYSTEM or CABLE concepts.

The knowledge base might be incomplete or inconsistent because constraints are missing. The domain model states which DESIGN-PROPERTIES have to be compared to which ENVIRONMENT-PROPERTIES. A constraint defines the relationship between these PROPERTIES. For example, while refining the domain model the expert constrained SAFE-ENERGY to be LESS EQUAL THREAT-ENERGY. KNACK looks for DESIGN-PROPERTIES or ENVIRONMENT-PROPERTIES that have no associated constraint.

```
What is the relationship between the SAFE
VOLTAGE property and the THREAT VOLTAGE.
property? less equal
```

The knowledge base might be incomplete or inconsistent because fixes are missing. KNACK assumes that fixes exist whenever a constraint is violated. If KNACK detects that a constraint has no associated fix, it indicates that to the expert. For example:

```
What design components represent a fix for the
THREAT VOLTAGE property? terminal protection
device
```

5. Rule Generation

KNACK stores the domain dependent knowledge it acquired from the expert in declarative form in its knowledge base. To create an expert system, this knowledge is proceduralized into OPS5 production rules [Forgy 81] using a simple parser written in LISP. These rules are then combined with domain independent rules representing the control knowledge.

The domain independent knowledge embodies the problem solving method. It establishes and controls the sequences of actions required to perform the evaluation task. This control knowledge dynamically defines the order in which subtasks have to be solved to perform the overall task. It also defines the knowledge roles that are applicable within each step. The problem solving methods described in the section 2 give some impression about the required control knowledge.

The domain dependent knowledge is organized in units according to the role that knowledge plays. The knowledge roles we have identified for the report-driven design evaluation task are:

- Strategy
- Constraint
- Fix
- Report Fragment
- Skeletal Report
- Information Identification
- Synonym

We will now describe the rules organized in these knowledge roles.

5.1. Strategy Rules

A WRINGER uses strategies to instantiate generalized concept characteristics with values describing a particular system design. The expert's input to define a question strategy was demonstrated with the following example:

```
Which strategy can be used to determine the
ENCLOSUREs of a SYSTEM?

[constant, question, inference, table, menu,
graphics, formula, database, postpone, quit]
[ QUESTION ]: <cr>

question text....: Please, list the enclosures

possible answers.. [ INCOMPLETE-SET, S-280C,
                   METAL BOX ] <cr>
default answer.... [ S-280C ]: unknown
status............ [ NOT-MANDATORY ]: <cr>
```

KNACK translates this description into three OPS5 rules. One rule (Rule 1) identifies that a question strategy can be used to gather a specific piece of information and asks the question. The second rule (Rule 2) checks whether a result of a strategy is valid, and the third rule (Rule 3) creates the slots for the strategy result(s).

```
Rule 1:

If   the goal is to identify strategies, and
     the subgoal is to determine the NAME of
         an ENCLOSURE, and
     a SYSTEM with some NAME is known,
then create a request to determine the NAME of
         an ENCLOSURE using a QUESTION
         strategy, and
     create the question "Please, list the
         enclosures of the <SYSTEM.NAME>
         system", and
     classify the answer as NOT-MANDATORY.
```

Whenever the WRINGER decides to determine the NAME of an ENCLOSURE, the above rule fires and establishes that a question strategy can be used to gather that piece of information. In case multiple strategies exists to determine the information similar rules would exist for each possible strategy. The WRINGER selects one strategy. If a question strategy is chosen, it asks the question.

The expert's input for the above defined question strategy further specifies that S-280C and METAL BOX are possible answers to that question. KNACK generates a rule to check the result of the strategy whether it is valid.

```
Rule 2:

If   the goal is to validate a strategy
         result, and
     the NAME of an ENCLOSURE was determined,
         and
     it is not S-280C or METAL BOX,
then mark the result as POSSIBLY INCORRECT.
```

This rule flags a strategy result as questionable because it is not a member of a predefined, incomplete set of possible answers. The WRINGER asks the user for confirmation of the result.

Once the WRINGER accepts the result of a strategy, it integrates the result with the already existing information.

```
Rule 3:

If    the goal is to integrate a strategy
          result, and
      the result is a value for the NAME of an
          ENCLOSURE, and
      a SYSTEM with some NAME is known,
then create a concept ENCLOSURE with a NAME
      characteristic, and
      instantiate it with that value, and
      create a link that the SYSTEM COMPRISES
          the ENCLOSURE.
```

Rule 3 creates the concept ENCLOSURE with a NAME characteristic and instantiates the concept characteristic with the strategy result. It further creates a relation linking the ENCLOSURE to the existing SYSTEM concept.

5.2. Constraint Rules

Constraint rules check the system design for violations of constraints imposed by a given environment. For example, the rule below determines whether the value for SAFE ENERGY is less equal than the value for THREAT ENERGY.

```
If    the goal is to identify constraint
          violations, and
      some THREAT ENERGY is known, and
      some SAFE ENERGY is known, and
      the SAFE ENERGY constrains the THREAT
          ENERGY, and
      the SAFE ENERGY is not less equal than
          the THREAT ENERGY,
then classify constraint 12 as VIOLATED.
```

5.3. Fix Rules

KNACK ensures that at least one fix exists for every constraint that can be violated. KNACK generates one rule for every potential fix.

```
If    the goal is to suggest a fix, and
      constraint 12 is VIOLATED,
then suggest a S-280C ENCLOSURE as a fix, and
      suggest a METAL BOX ENCLOSURE as a fix.
```

This rule suggests a S-280C and a METAL BOX ENCLOSURE as a possible fixes for the violated constraint. The WRINGER checks which of the possible fixes will satisfy the constraint gathering additional information if required. It then suggests that fix to the designer. If the designer agrees, the WRINGER integrates the enclosure with the existing design and updates the design parameters.

5.4. Report Fragment Rules

Report fragment rules represent the content of a WRINGERs report.

```
If   the goal is to print the report, and
     report fragment 9 can be selected, and
     an ENVIRONMENT with NAME EMP is known,
         and
     an ENCLOSURE with some NAME is known,
then print "*LOOPOVER* <ENCLOSURE.NAME>
         @SECTION [ Shielding Requirements for
         the <ENCLOSURE.NAME> ENCLOSURE ]".
```

To present the report in an appealing format (include headings, tables, etc) the output of a WRINGER is formated by a text formatting program. For that reason, the print action of the rule contains commands for the text formatting program, in our example @SECTION.

Each report fragment from the sample report is proceduralized by one or more OPS5 rules. Fragments that were generalized to contain a list of representatives for the same concept need additional control to realize the intended repetitions of parts of the fragment. Multiple OPS5 rules are necessary to control the repetitions: one rule to identify the variable denoting the list of representatives, one rule to print the text preceding the repetition once, one rule to print the repetition for each instantiation of the variable, and one rule to print the text succeeding the repetition once. The same principle applies to chapter, section, and subsection headings that contain variables. Rules are added that control the repetition of an entire chapter, section, or subsection for each instantiation of the variable. The term "*LOOPOVER* <ENCLOSURE.NAME>" in the above example is a command for the LISP parser to create those control rules. KNACK inserts a corresponding "*LOOPEND*" as the last word of the chapter, section, or subsection.

5.5. Skeletal Report Rules

Skeletal report rules represent the outline of the report a WRINGER produces. As indicated in the sample interaction, KNACK will insert a new chapter, section or subsection whenever it discovers a keyword like CHAPTER, SECTION, or SUBSECTION in a report fragment. An OPS5 rule representing a part of the skeletal report is created for each chapter, section, and subsection heading. The skeletal report rule for the section heading of the above example is shown below in an english translation.

```
If   the goal is to create the skeletal
         report,
then create section 2 of chapter 1
     with the heading "Shielding Requirements
         for the Enclosures", and
     establish that fragment 9 can be
         selected, and
     establish that fragment 10 can be
         selected, and
     establish that fragment 11 can be
         selected, and ...
```

The rule defines the section heading as it appears in the table of contents of the report the WRINGER is trying to produce. It also specifies the fragments that can be selected into the section and determines the order of the fragments.

5.6. Information Identification Rules

As indicated in section 2 a WRINGER collects related information in a coherent manner by following the outline of its report. KNACK generates a rule for each chapter, section, or subsection determining the information relevant to that report part. The following example shows part of the information identification rule for the section "Shielding Requirements for the Enclosures" in an english translation.

```
If   the goal is to determine an existing
         design, and
     the current report part is chapter 1,
         section 2,
then create the subgoal to determine the NAME
         of an ENVIRONEMNT, and
     create the subgoal to determine the NAME
         of an ENCLOSURE, and
     create the subgoal to determine the NAME
         of an APERTURE, and
     create the subgoal to determine the
         MATERIAL of an ENCLOSURE, and
     create the subgoal to determine the
         THICKNESS of an ENCLOSURE ...
```

5.7. Synonym Rules

Synonym rules provide a simple mechanism to deal with varying or conflicting terminology of different designers. They are implemented as demons. Whenever the designer interacting with a WRINGER uses a term known to be a synonym for some basic expression, the synonym is being transformed into the basic expression. The rules are simple:

```
If   the NAME of a CABLE was determined, and
     is ANTENNA CABLE
then change it to SIGNAL CABLE.
```

6. KNACK's Scope

KNACK derives its power by exploiting a presupposed problem-solving method. The method explicates the types of knowledge (knowledge roles) needed to solve tasks in a particular domain. The underlying assumption is that a problem-solving method and the associated knowledge roles cover a number of tasks in a particular domain. To get a better understanding of the kinds of tasks KNACK's assumed method can solve, KNACK has been and is being used by knowledge engineers to create a series of evaluation systems. The following describes these tasks, the experience gained, and some data on KNACK's performance and scope.

6.1. KNACK Tasks

KNACK was used to generate an initial knowledge base for a number of expert systems. The expert systems were then manually enhanced to accommodate the specific demands of a particular task. The enhancements uncovered deficiencies and shortcomings in KNACK's approach to acquire knowledge, assumed problem-solving method, and associated knowledge roles. KNACK has been improved to address the problems and is now being used to re-generate the initial systems.

The following application systems are being created with KNACK:

The Data Item Description WRINGER Family: The DPR WRINGER, used as an example throughout this paper, is a member of the Data Item Description WRINGER Family. Three WRINGERs have been developed for very similar tasks. They assist with different stages in the design of electromechanical systems for the nuclear hardening domain. Nuclear hardening involves the use of specific engineering design practices to increase the resistance of an electromechanical system to the environmental effects generated by a nuclear weapon. Designers of electromechanical systems usually have little or no knowledge about the specialized analytical methods and engineering practices of the hardening domain. The purpose of the WRINGERs is to assist a designer in improving given designs of electromechanical systems that may be suboptimal from a hardening perspective. The WRINGERs assume that the initial design describes a technically functional system. They evaluate the design from a hardening perspective. The suggested improvements are either extensions to the design or recommendations for using different design components. The WRINGERs present the design, together with the results of the evaluation, in the form of a technical document that meets government requirements.

The first WRINGER, a PROGRAM PLAN writer, evaluates and presents the primary top level report covering all phases of the design project. It took 7 person-days to create the WRINGER with KNACK. Its knowledge base contains 795 OPS5 rules. The second expert system, a DESIGN PARAMETERS REPORT writer, evaluates and presents a detailed description of an electromechanical system ranging from the level of major components to the level of individual semiconductors. It took 21 person-days to create the WRINGER with KNACK. Its knowledge base contains 1446 OPS5 rules. The last WRINGER, a TEST PLAN writer, produces a plan to confirm the hardness of a design. This includes a list of the design components to be tested, a description of the tests to be performed, and the expected test results. It took 8 person-days to create the WRINGER with KNACK. Its knowledge base consists of 230 rules. The WRINGERs were created with a previous implementation of KNACK, reported in [Klinker 87b], and are now being tested by the organization that will use them. That previous implementation required from the expert to explicitly define the generalized report and strategies. The experience gained with that task led to a refinement of KNACK's approach to acquire knowledge: the introduction of the domain model and the automation of the generalization process.

The XNET-Design WRINGER: A WRINGER is being developed to assist a sales-person with the design and configuration of computer networks. In general, a sales-person has a good understanding about aspects like costs, compatibility, and extensibility when he is designing a network which suits his customer best. But usually he has little or no knowledge about the technical aspects involved. The purpose of the WRINGER is to assist a sales-person in improving his design of computer networks that may be suboptimal from a technician's perspective. The WRINGER assumes that the initial design describes the computing environment but might not be a technically functional computer network. It evaluates the design description from a technician's perspective. The suggested improvements are either extensions to the design or recommendations for different interconnections between design components. The WRINGER's output is a list of generic network components and their interconnections serving as input for a program that will select the specific parts. The WRINGER is in the very first stage of development. Data that describe KNACK's performance are not yet available. The experience gained with that task led to a refinement of KNACK's assumed problem-solving method: The XNET-Design WRINGER focuses on designing a computer network rather than on refining an existing network. The WRINGERs of the Data Item Description WRINGER Family first require to describe an existing system and then evaluate that system from a hardening perspective, asking the designer to confirm the suggested fixes. That distinction is not applicable for the XNET-Design WRINGER. It takes whatever input the sales-person can give and completes the design, applying fixes without asking for confirmation. A switch has been introduced into a WRINGERs problem-solving method that allows a designer to choose between the functions "gather information and evaluate" and "gather information and complete"

The Software Requirements WRINGER: A WRINGER is being developed to assist a systems analyst with the definition of requirements for software. Defining requirements for new software is a very complex process. It involves functionally decomposing the software into basic modules, defining the data requirements, and integrating the new software with the existing software environment. One systems analyst alone might not have enough knowledge about the existing software environment. The purpose of the WRINGER is to assist a systems analyst in refining the requirements for software systems that may be suboptimal given the existing software environment. The WRINGER assumes that the initial design describes the new software on a high level of abstraction. It supports the systems analyst in functionally decomposing that description into basic modules and defining the data requirements for the modules. The WRINGER evaluates the design as to whether it is compatible with the existing software environment. The suggested improvements are refinements to the requirements of the new software. The WRINGER produces a technical document describing the requirements for the software system. The document further contains an executive summary with an opinion about whether the new software will be a valuable enhancement of the existing software. It took 18 person-days to create the WRINGER with KNACK. Its knowledge base consists of 291 rules. The experience gained with that task led to the introduction of two new strategies to acquire information: It is critical for the Software Requirements WRINGER to support a user with the decomposition of software requirements. Simple graphics allow to represent requirements for software and data in the form of nodes and the data flow in the form of directed links between nodes. Forms are used to describe the nodes further.

The Project Progress Report WRINGER: A WRINGER is being developed to assist a project leader with the assessment of a project's progress. A project leader might not have enough experience to create a project plan and assess the progress of a project. Also, he might not have enough knowledge to integrate his project into the broader objective of his management. The purpose of the WRINGER is to assist a project leader in refining a project plan that may be suboptimal from a management perspective. The WRINGER assumes that an initial proposal can be provided. It supports the project leader in creating and updating the project plan according to the progress of the project. The WRINGER evaluates the plan from a management perspective. The suggested improvements are refinements or changes to the plan. The WRINGER produces a proposal, project plan, and periodical progress reports that will allow management to assess the progress of a project. It took 19 person-days to create the WRINGER with KNACK. Its knowledge base consists of 429 rules. The experience gained with that task led to an extension of KNACK's build in knowledge of evaluation: For the Data Item Description WRINGER Family the interdependencies between the concepts describing a design and an environment are appropriately represented by a tree structure. The Project Progress Report WRINGER requires a network.

The Business Plan WRINGER: A WRINGER is being developed to assist an entrepreneur in the preparation of business plans. The first step in creating a business is to seek investment capital. For this purpose, entrepreneurs generate business plans. A business plan contains information on the planned business, e.g. the industry, the product, the market and marketing plan, production, personnel, and financial projections. An entrepreneur usually has little knowledge about how to create a business plan. The purpose of the WRINGER is to support an entrepreneur in developing a business plan to secure investment capital from venture capitalists. The WRINGER assumes that the entrepreneur can provide a description of the product and the goals of the proposed business. It will then support the entrepreneur in creating the business plan. The WRINGER evaluates the plan from a venture capitalist's perspective. It produces a document containing the necessary details and justifications to demonstrate the proposed business to an investor. The information is presented in a way appealing to an investor. The WRINGER is in the very first stage of development. Data that describe KNACK's performance are not yet available. The experience gained with that task led to an extension of a WRINGER's representation of acquired information: The Business Plan WRINGER requires a temporal dimension. The same types of data project a planned business in different, succeeding time periods.

Each application improved our understanding of the evaluation task and had some implications on the development of the KNACK tool. We continue to improve KNACK. The goal is that a domain expert can use the tool to generate all of the knowledge bases for the above described WRINGERs. The different applications gave us some insight into KNACK's scope. At a first glance, those applications seem to be quite different. But a closer look reveals that they all meet some common requirements. The task is constructive evaluation, i.e., an existing plan or design has to be evaluated to determine whether it meets additional constraints not anticipated in the original design. The evaluation is constructive because fixes can be suggested in case of constraint violations. The use of KNACK imposes some requirements on the expert: The expert must be able to provide an initial model of the domain, he must be able to express some of his knowledge in the form of a sample report, and he must be able to define strategies that a WRINGER can use to instantiate concepts with values describing a particular plan or design. Finally, the use of a WRINGER requires that a designer can provide an initial design.

6.2. Some Performance Data

This section gives some impression of KNACK's performance in creating the Program Plan WRINGER (PP WRINGER), Design Parameters Report WRINGER (DPR WRINGER), Test Plan WRINGER (TP WRINGER), Software Requirements WRINGER (SR WRINGER), and Project Progress Report WRINGER (PPR WRINGER). The Business Plan WRINGER and XNET-Design WRINGER are in the very first stages of development. Data that describe KNACK's performance in creating them are not yet available. The data of Table 6-1 describe the complexity of the domains, Table 6-2 contains some data about the complexity of the generated knowledge bases, and Table 6-3 summarizes the effort involved in creating the expert systems with KNACK.

Table 6-1 gives some impression of the complexity of the domains for the five WRINGERs. It describes the input the experts had to provide in terms of the sample report, the domain model, and the sample strategies. Strategies can be interactive, i.e., they elicit information from the WRINGER users, or autonomous, i.e., they infer information based on previously provided information. Examples of interactive strategies are: questions, graphical design descriptions, menus, forms, or tables. Examples of autonomous strategies are: inferences, database lookups, or formulas.

Table 6-2 describes the generated knowledge base for the five WRINGERs. The size of the knowledge base is determined by the number of OPS5 rules it contains. The conditionality of a rule is described by the number of its condition elements. Each condition element can be instantiated by a concept. The complexity of a condition element is defined by the number of characteristics which describe a concept. The action part of a rule is described by the number of actions the rule performs. An action either creates a new concept or modifies an existing concept characteristic.

Finally, Table 6-3 gives some impression of the time involved to generate a WRINGER using KNACK. Creating WRINGERs is an iterative process. Whenever a WRINGER reveals inadequacies, KNACK is used to improve it. Table 6-3 shows the time spent to generate the initial knowledge bases for the WRINGERs described in section 6.1 and in the tables 6-1 and 6-2. This includes the effort for the initial

	PP	DPR	TP	SR	PPR
Number of fragments in the sample report	237	455	88	203	113
Average size of each fragment in words	9.5	14.7	14.3	10.2	8.1
Number of concepts	43	92	28	55	109
Average number of characteristics for each concept	2.3	3.7	2.5	3.6	1.4
Number of interactive strategies	72	152	35	21	92
Number of autonomous strategies	22	159	4	32	41

Table 6-1: Complexity of the Domain

	PP	DPR	TP	SR	PPR
Number of rules	795	1446	230	291	429
Average number of condition elements per rule	3.4	3.8	1.1	1.9	2.9
Number of characteristics per condition element	2.0	2.3	1.5	2.0	2.5
Number of actions per rule	7.2	3.0	2.9	4.1	1.9

Table 6-2: The Knowledge Base

input (sample report, domain model, and sample strategies), the generalization process (sample report and sample strategies), and the review of the knowledge base. The effort for the generalizations includes the expert's corrections to the sample instantiations of the generalized fragments and strategies. Since the PP WRINGER, DPR WRINGER, and the TP WRINGER were created with a previous implementation of KNACK, no detailed data are available.

7. Conclusion

Existing expert systems have proven that AI techniques can be used to solve a variety of knowledge intensive problems. But expert systems are time-consuming to develop and difficult to maintain. A key issue in developing any expert system is how to update its large and growing knowledge base. It has been shown that a large knowledge base can be kept maintainable by organizing it according to the different roles that knowledge plays [Chandrasekaran 83], [Clancey 83], [Neches 84]. Based on this realization a variety of knowledge acquisition tools have been produced during the past years to overcome those development and maintenance problems.

Existing knowledge acquisition tools focus on different aspects of the knowledge engineering task. KREME [Abrett 87] provides an environment for editing large knowledge bases. SEAR [van de Brug

	PP	DPR	TP	SR	PPR
Number of days to create the sample report	--	--	--	2	3
Number of days to create the preliminary domain model	--	--	--	4	5
Number of days to create the sample strategies	--	--	--	3	1
Number of days to generalize the sample report	--	--	--	4	5
Number of days to generalize the sample strategies	--	--	--	4	5
Number of days to review the knowledge base	--	--	--	1	--
Total	7	21	8	18	19

Table 6-3: Effort

86], AQUINAS [Boose 87], KRITON [Diederich 87], and TKAW [Kahn 87] integrate a variety of methodologies and tools for the development of expert systems into a workbench for a knowledge engineer.

Other knowledge acquisition tools try to automate the knowledge acquisition process. An automated knowledge acquisition tool typically interacts with domain experts directly. No knowledge engineer is necessary to translate the expert's knowledge into production rules. An automated knowledge acquisition tool further organizes the knowledge it acquires, and generates an expert system. The domain expert can also use it to test and maintain the program it generates. The critical feature of such a tool is that a domain expert can use it without having to know about programming in general and specific AI techniques. Examples of automated knowledge acquisition tools are TEIRESIAS [Davis 82], ETS [Boose 84], MORE [Kahn 85], MOLE [Eshelman 87], and SALT [Marcus 87]. These tools derive their power by presupposing the problem solving method of the expert systems they generate [McDermott 86], [Gruber 87]. Other automated knowledge acquisition tools like OPAL [Musen 87] and STUDENT [Gale] exploit an explicit domain model.

A useful distinction between the above knowledge acquisition tools is whether they help to create expert systems that either select or construct a solution [Clancey 84]. TEIRESIAS, ETS, MORE, MOLE, OPAL and STUDENT generate expert systems that select a solution from a given set of pre-enumerable candidates. SALT is an example of a knowledge acquisition tool for systems that construct a solution.

This paper described KNACK, a knowledge acquisition tool that generates expert systems for evaluating different classes of designs. Like SALT, it can be used to develop expert systems that construct a solution compatible with a set of constraints. But whereas SALT generated expert systems produce designs from scratch, i.e., typically one designer has complete knowledge about all constraints a solution has to satisfy, KNACK generates evaluation systems. Evaluation systems assume that multiple designers are involved in a design task and each designer only knows a subset of the constraints a solution has to satisfy.

Another difference between KNACK and the tools mentioned so far is KNACK's report-driven approach to acquiring knowledge. KNACK assumes that an expert can present his knowledge adequately in the form of a report. The expert must have a clear understanding of what constitutes an acceptable report describing and evaluating a design. This includes that the expert knows what information is needed, how to evaluate this information, and how a designer should present this information.

If categorized along the dimensions outlined above, KNACK exploits a presupposed problem solving method as well as an explicit domain model. Like TEIRESIAS, ETS, MORE, MOLE, SALT, and SEAR it presupposes and exploits the problem-solving methods and the knowledge roles of the expert systems it generates. Like OPAL and STUDENT, KNACK exploits a domain model during knowledge acquisition. KNACK uses the domain model to elicit knowledge in a format familiar to the expert and develop expectations about the knowledge the expert might provide. KNACK differs from OPAL and STUDENT in that the domain model can be customized for a particular domain and no knowledge engineering expertise is required to build a domain model. Also, KNACK does not assume that its domain model is complete and consistent. It expects that the expert can provide a preliminary model and gradually augments that preliminary model during knowledge acquisition into a domain model describing the design and the evaluation process.

The description of KNACK's approach for knowledge acquisition reveals that the generalization process is critical for KNACK's performance. The technique uses simple heuristics to replace fixed text components of the sample report with variables denoting concepts and concept characteristics. The heuristics take into account the structure of the sample report and previous generalizations. They look for keywords, known concepts, and concept representatives in the sample report. Since the concepts and concept representatives are described in the domain model, the domain model is the heart of the generalization process. A successful generalization requires that the concepts and concept representatives in the domain model closely correspond to the terms used in the sample report. Moreover, the domain model must contain a detailed structural and functional description of the evaluation task. We will continue to make KNACK less sensitive to incomplete domain models. This includes getting a better understanding of the knowledge common to a range of evaluation tasks and improving the heuristics used in the generalization process. We will further extend KNACK's ability to use the expert's corrections as well as the generalized strategies to refine the preliminary model into a structural and functional representation of a particular evaluation domain.

An important goal in our research is to make the use of KNACK independent from knowledge engineering expertise. While we believe that KNACK's approach described in this paper is an important step towards that goal, we do realize that we are only a short way along the path.

Acknowledgments

Many people made significant contributions to the KNACK project. First comes John McDermott who helped shape the direction of the project from the beginning. Joel Bentolilia, Casey Boyd, Gilbert Caplain, David Dong, Serge Genetet, Michael Grimes, Don Kosy, Jon Maiman, Beatrice Paoli-Julliat, Bob Schnelbach, and Martin Stacey were or still are members of the group. William Rodi (S-Cubed) also made important contributions. Thomas Flory and Roland Polimadei of Harry Diamond Laboratories (HDL), Rodney Perala of Electro Magnetic Applications (EMA), and Don Vincent of Booz-

Allen&Hamilton served as our domain experts. We would also like to thank Andrej Bevec (HDL), John Northrop (S-Cubed), William Proffer (S-Cubed), and Alex Stewart (HDL) for their support. Sandra Marcus provided very helpful comments on an earlier draft of this paper.

References

[Abrett 87] Abrett, G., and M. Burstein.
The KREME Knowledge Editing Environment.
International Journal of Man-Machine Studies (), to appear 1987.

[Boose 84] Boose, J.
Personal construct theory and the transfer of human expertise.
In *Proceedings of the National Conference on Artificial Intelligence.* Austin, Texas, 1984.

[Boose 87] Boose, J., and J. Bradshaw.
Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge Acquisition Workbench for Expert Systems.
International Journal of Man-Machine Studies 26(1):3 - 28, 1987.

[Chandrasekaran 83]
Chandrasekaran, B.
Towards a taxonomy of problem solving types.
AI Magazine 4(1):9-17, 1983.

[Clancey 83] Clancey, W.
The advantages of abstract control knowledge in expert system design.
In *Proceedings of the 3rd National Conference on Artificial Intelligence.* Washington, D.C., 1983.

[Clancey 84] Clancey, W.
Classification problem solving.
In *Proceedings of the 4th National Conference on Artificial Intelligence.* Austin, Texas, 1984.

[Davis 82] Davis, R. and D. Lenat.
Knowledge-Based Systems in Artificial Intelligence.
McGraw-Hill, 1982.

[Diederich 87] Diederich, J., I. Ruhmann, and M. May.
KRITON: A Knowledge Acquisition Tool for Expert Systems.
International Journal of Man-Machine Studies 26(1):29 - 40, 1987.

[Eshelman 87] Eshelman, L., D. Ehret, J. McDermott, and M. Tan.
MOLE: A Tenacious Knowledge Acquisition Tool.
International Journal of Man-Machine Studies 26(1):41 - 54, 1987.

[Forgy 81] Forgy, C.L.
OPS5 user's manual.
Technical Report, Carnegie-Mellon University, Department of Computer Science, 1981.

[Gale] Gale, W.
Knowledge Based Knowledge Acquisition for a Statistical Consulting System.
International Journal of Man-Machine Studies 26(1):55 - 64, .

[Gruber 87]	Gruber, T., and P. Cohen. Design for Acquisition: Principles of Knowledge System Design to Facilitate Knowledge Acquisition. *International Journal of Man-Machine Studies* 26(2):143 - 159, 1987.
[Kahn 85]	Kahn, G., S. Nowlan and J. McDermott. MORE: an intelligent knowledge acquisition tool. In *Proceedings of Ninth International Conference on Artificial Intelligence*. Los Angeles, California, 1985.
[Kahn 87]	Kahn, G., E. Breaux, R. Joseph, and P. DeKlerk. An Intelligent Mixed-Initiative Workbench for Knowledge Acquisition. *International Journal of Man-Machine Studies* (), to appear 1987.
[Klinker 87a]	Klinker, G., C. Boyd, S. Genetet, and J. McDermott. A KNACK for Knowledge Acquisition. In *Proceedings of Sixth National Conference on Artificial Intelligence*. Seattle, Washington, 1987.
[Klinker 87b]	Klinker, G., J. Bentolila, S. Genetet, M. Grimes, and J. McDermott. KNACK - Report-Driven Knowledge Acquisition. *International Journal of Man-Machine Studies* 26(1):65 - 79, 1987.
[Marcus 87]	Marcus, M. Taking Backtracking with a Grain of SALT. *International Journal of Man-Machine Studies* 26(4):383 - 398, 1987.
[McDermott 86]	McDermott, J. Making expert systems explicit. In *Proceedings of 10th Congress of the International Federation of Information Processing Societies*. Dublin, Ireland, 1986.
[Musen 87]	Musen, M., L. Fagan, D. Combs and E. Shortliffe. Using a Domain Model to Drive an Interactive Knowledge Editing Tool. *International Journal of Man-Machine Studies* 26(1):105 - 121, 1987.
[Neches 84]	Neches, R., W. Swartout, and J. Moore. Enhanced maintenance and explanation of expert systems through explicit models of their development. In *Proceedings of IEEE Workshop on Principles of Knowledge-based Systems*. Denver, Colorado, 1984.
[van de Brug 86]	van de Brug, A., J. Bachant, J. McDermott. The Taming of R1. *IEEE Expert* 1(3), 1986.

Table of Contents

List of Tables