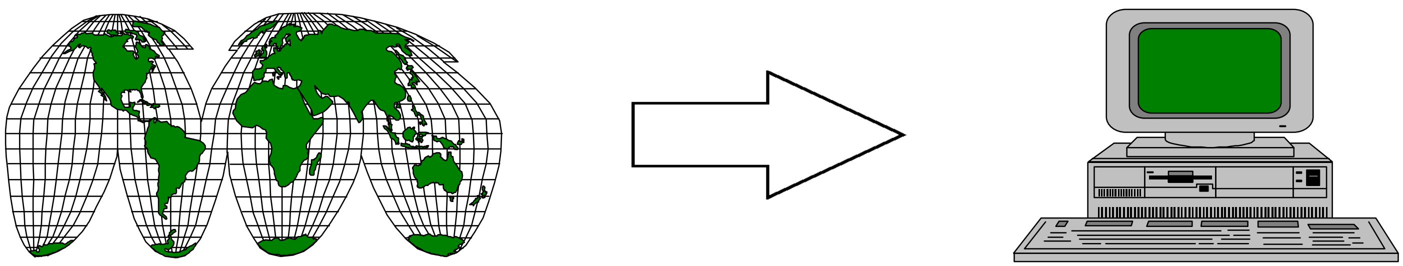
Agenda

- <u>Knowledge engineering concepts</u>
- **Current trends in knowledge-based development**
- **Break**
- **Case Studies**
- **Incorporating knowledge engineering tools into software projects**
- **Summary: Lessons learned and future directions**
- **Questions**

Knowledge Engineering Concepts

- Definition of knowledge engineering
- The challenge of knowledge acquisition
- Basic concepts and terminology
- Approaches for knowledge engineering

What Is Knowledge Engineering?



K **K***

Knowledge engineering is the acquisition, management, and processing of knowledge to produce systems that assist and support human activities

The Facets of Knowledge Engineering

- **Acquisition:**
  - » **Transformation of knowledge from the forms in which it is available in the world into forms that can be used by a knowledge system**
  - » **Deals with knowledge representation issues**
- **Management**
  - » **Organization, consistency and maintenance of acquired knowledge**
- **Processing**
  - » **Execution of solutions and explanations**

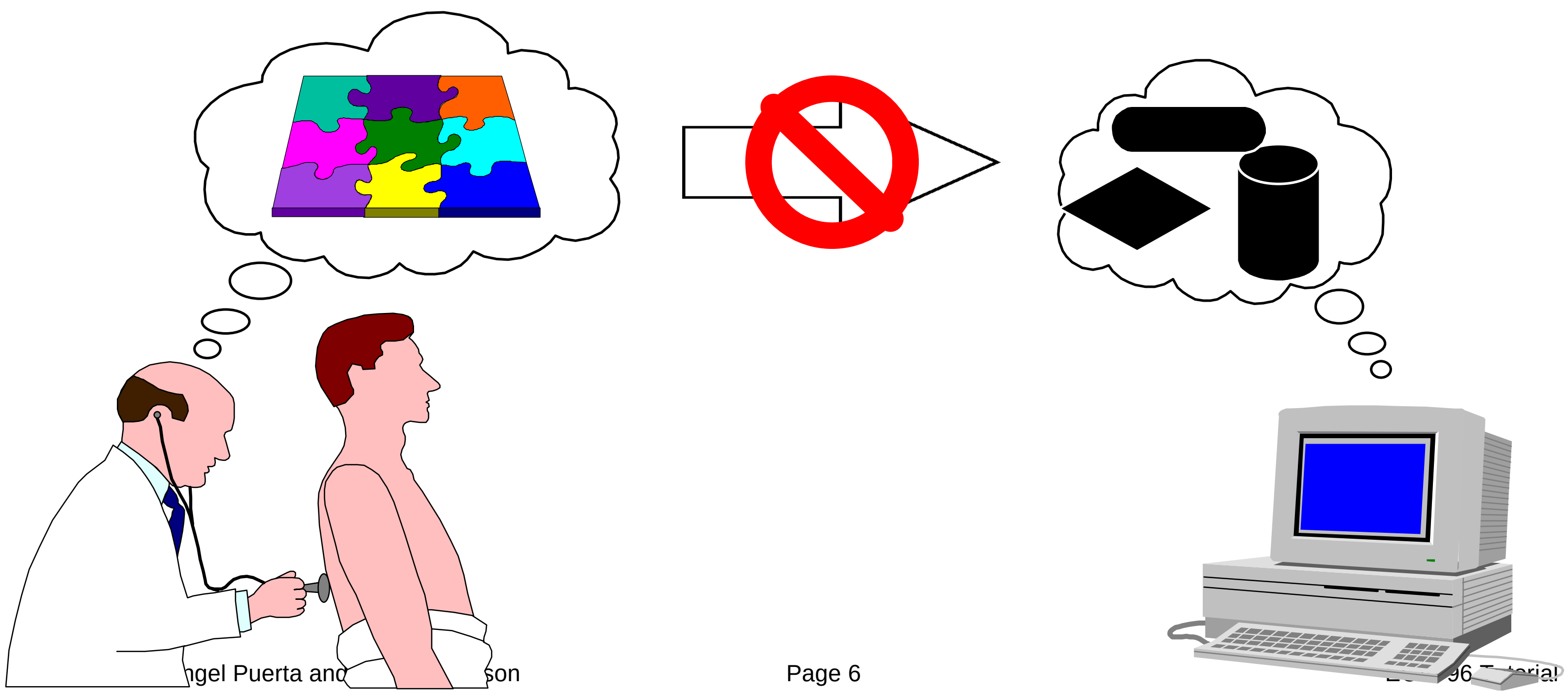
We will emphasize knowledge acquisition issues during the tutorial

The Knowledge Acquisition Bottleneck

- **Nothing happens until knowledge is acquired**
- **Expert system shells support mostly maintenance and processing**
- **Sources of knowledge are unreliable**
  - » **Domain experts provide <u>incomplete</u>, even <u>incorrect</u> knowledge**
  - » **Domain experts may not be able to articulate their knowledge**
- **Knowledge bases are hard to build**
  - » **Computational knowledge representations are complex**

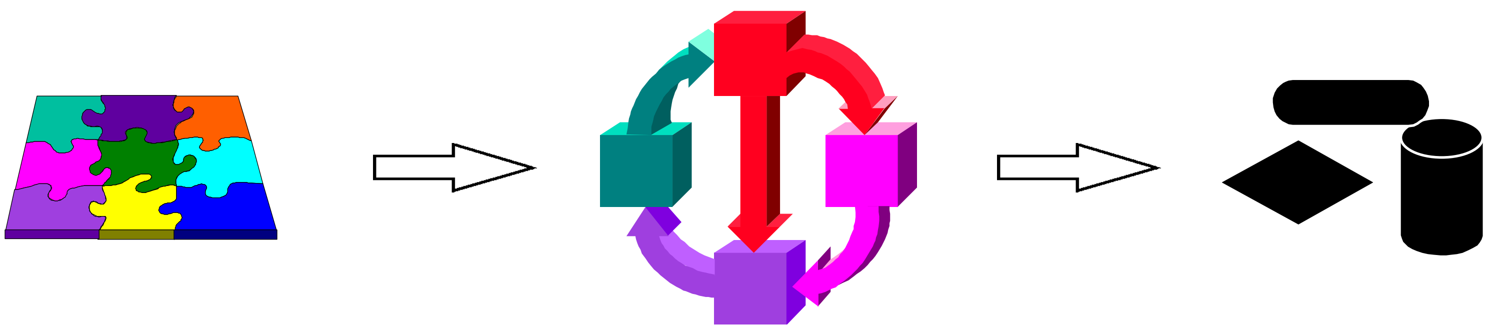
Acquiring Expert Knowledge: Transferring versus Modeling

It is not possible to transfer directly a domain’s expert knowledge to a machine because the respective representations are too dissimilar



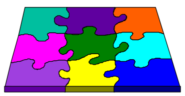
Acquiring Expert Knowledge: Transferring versus Modeling (2)

Knowledge acquisition is a <u>modeling</u> process. A knowledge engineer builds a <u>theory</u> of a domain and then makes that theory <u>operational</u>

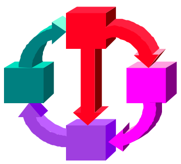


Domain Knowledge | **Domain Theory** | **Operational Theory**

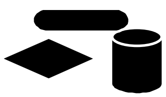
Steps to Engineer Knowledge



- **Conduct task analysis**
  - » **Elicit knowledge from people**
  - » **Elicit knowledge from observations**



- **Conduct knowledge-level analysis**
  - » **Build a representation of the task and its domain in an appropriate language**



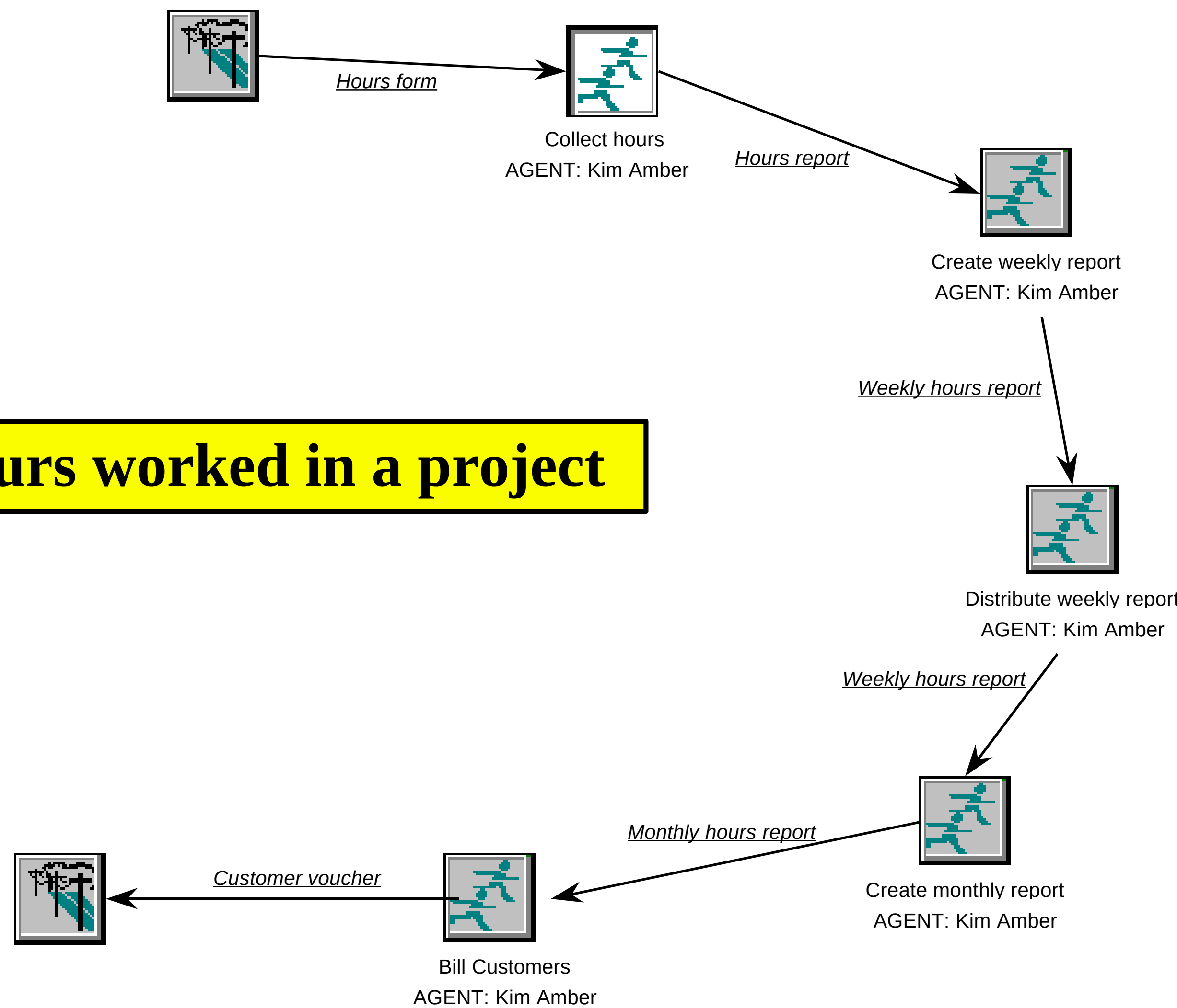
- **Operationalize formal representations**
  - » **Build a machine-executable representation**
  - » **Maintain and process knowledge base**
- **Iterate, iterate, iterate**

Task Analysis

- A task analysis produces a model of the task to be automated
- A task analysis normally requires a workplace analysis
- Task analysis techniques:
  - » Interviews with domain experts
  - » Interviews with people affected by the task
  - » Observations of people performing the task
  - » Videotaping of people performing the task

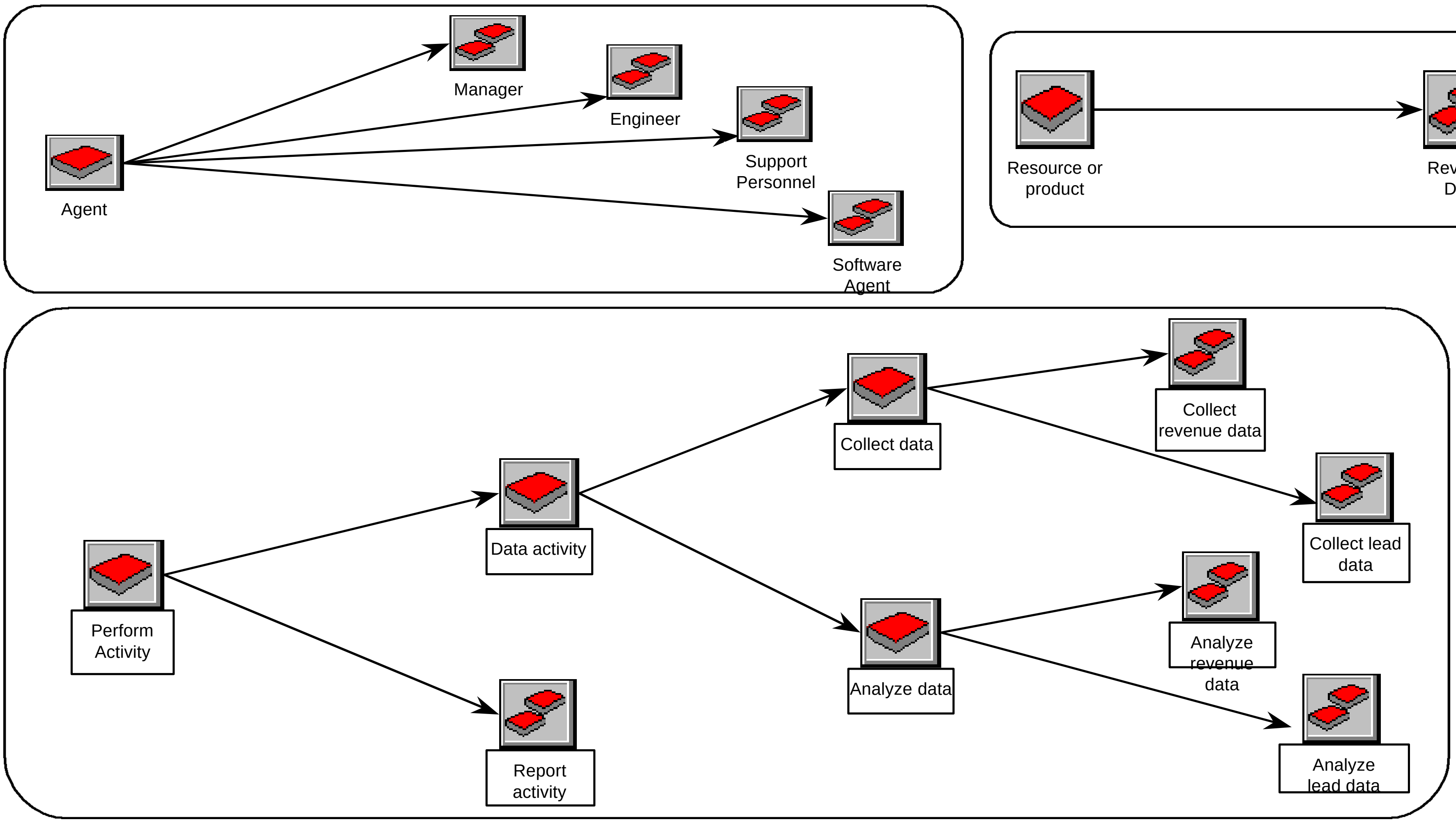
Task Analysis Example

Task: Billing hours worked in a project



Task Analysis Example (2)

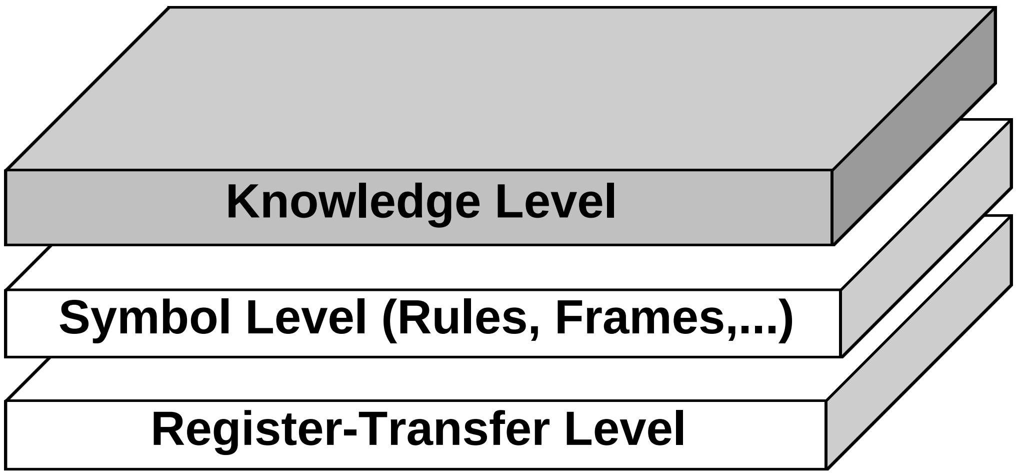
Multiple levels of abstraction allow multiple views on the task



Knowledge-Level Analysis

A knowledge-level analysis of a system produces a description of the behavior of that system without any assumptions about the knowledge representation that such system will use

(Newell, 1982)



Structure of the Knowledge Level

- **A knowledge-level description consists of**
  - » Agents
  - » Environment
  - » Actions
  - » Body of knowledge
  - » Goals

An intelligent <u>agent</u> working within a specific <u>environment</u> uses its <u>body of knowledge</u> to select <u>actions</u> that can achieve the agent's <u>goals</u>

Knowledge-Level Representations

- A language must exist to talk about knowledge-level descriptions
- The language must make no assumptions about knowledge representation
- Examples of knowledge-level languages:
  - » Logic
  - » Natural Language
  - » KADS
  - » Models of problem-solving methods

The KADS Modeling Approach

Four-layer model. Relations link one layer to another.

Level	Relations	Objects	Organization
Domain		Concepts, Relations Structures	Axiomatic Structure
	Describe		
Inference		Meta-classes, Knowledge Sources	Inference Structure
	Apply		
Task		Goals, Tasks	Task Structure
	Control		
Strategy		Plans, Meta-rules, Repairs, Impasses	Process Structure

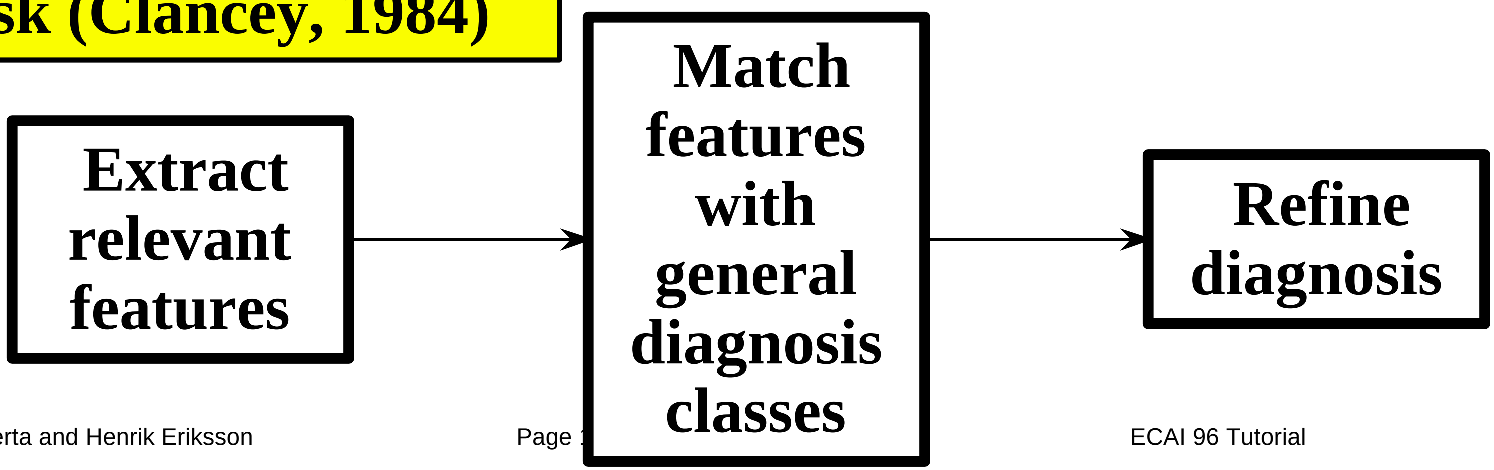
(Wielinga, 1992)

Models of Problem-Solving Methods

- **A model of a problem-solving method defines *how* a task will be accomplished by the system**
  - »**Uses knowledge-level terms (actions, goals,...)**
  - »**Is domain independent**
  - »**Makes no commitment to particular knowledge-representation languages**

(McDermott, 1988)

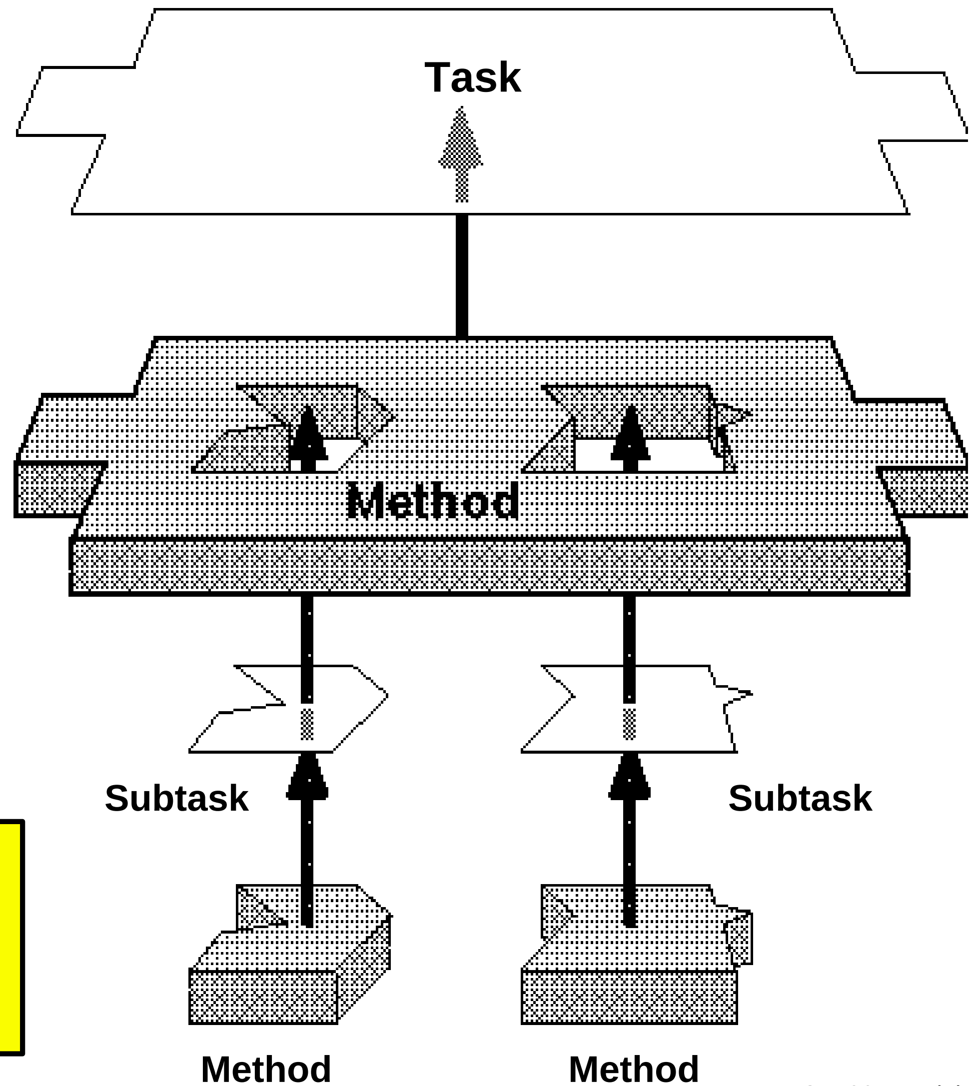
Example: Heuristic classification method for a diagnosis task (Clancey, 1984)



Tasks and Methods

Task = <u>what</u> to do

Method = <u>how</u> to do it



Task and Methods are the basis of reusable knowledge components

Operationalization

- The products of the knowledge-level analysis must be made <u>executable</u>
  - » A symbol-level representation must be chosen (i.e., rules, frames,...)
  - » Knowledge-level models must be written in the selected representation

After knowledge acquisition a system contains:

- **Knowledge about the domain**
- **Knowledge about the task**
- **Knowledge about solving the task**

Reusable Knowledge Components

- *Chunks* of knowledge that:
  - » Can be used in more than one knowledge-based system
  - » Do not require significant modifications before reuse
  - » Can be combined in a predefined manner with other reusable components
- Examples
  - » A method to solve room-assignment problems
  - » A domain model of clinical trials
- Reusability is critical in knowledge engineering because of high development costs

Review: Knowledge Engineering Concepts

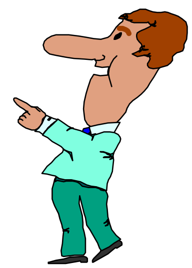
Knowledge engineering means acquiring, managing, and processing knowledge

Knowledge engineering steps:

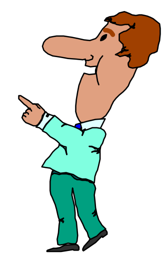
» **Task analysis**

» **Knowledge-level analysis**

» **Operationalization**



Knowledge acquisition is the most critical phase of knowledge engineering



Agenda

- Knowledge engineering concepts
- Current trends in knowledge-based development
- Break
- Case Studies
- Incorporating knowledge engineering tools into software projects
- Summary: Lessons learned and future directions
- Questions

Overview

- Review of knowledge-acquisition tools
- Metatools for knowledge acquisition
- Knowledge-engineering environments for reusable components
- Internet-based approaches

Review of Knowledge-Acquisition Tools

- **Goals:**
  - » **To gain a historical perspective on the development of KA tools**
  - » **To understand the current design trends in knowledge-acquisition tools**
- **Knowledge acquisition tools**
  - » **Symbol-level tools**
  - » **Model-formulation tools**
  - » **Method-specific tools**
  - » **Domain-specific tools**
- **Maintenance and processing tools**
- **Knowledge-engineering environments**

KA Symbol-Level Tools

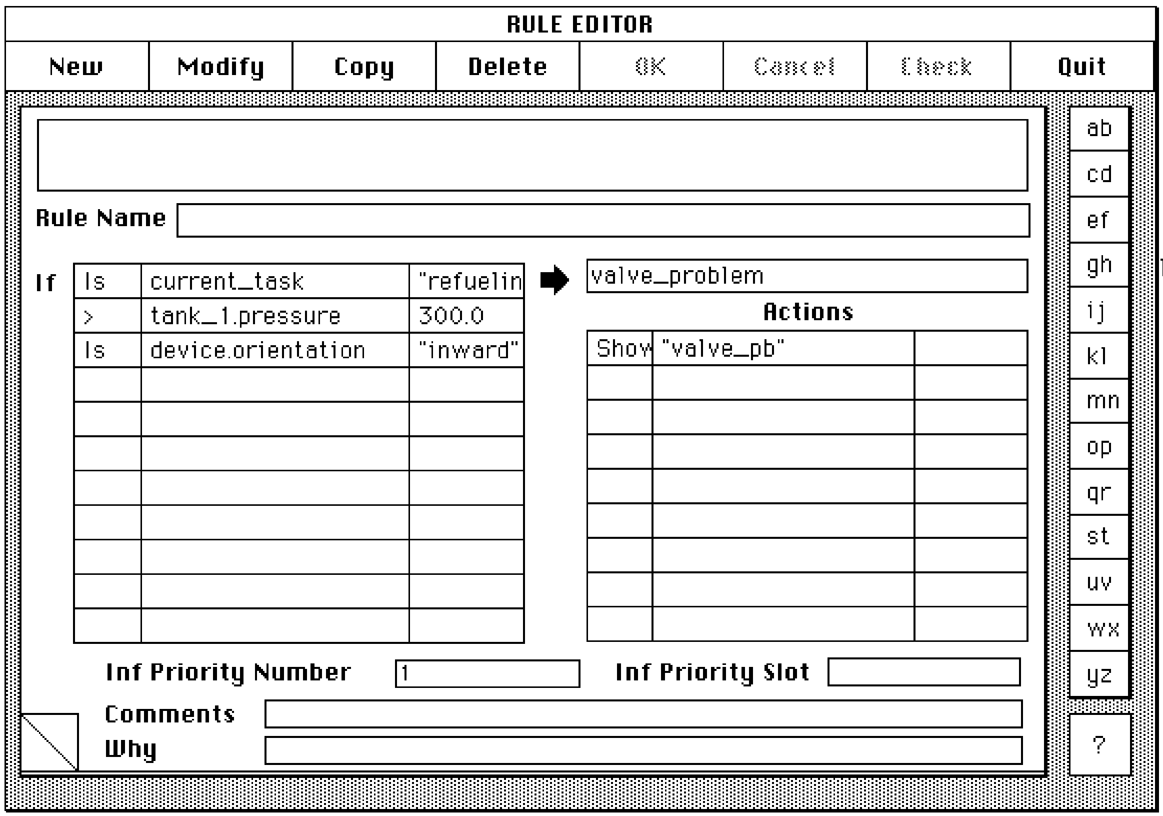
- Require users to manipulate symbols (e.g., rules) to build expert systems
- Ignore issues of knowledge-level analysis
- Low level of abstraction
- Examples
  - » Rule editors
  - » Frame editors
  - » Diagram editors
  - » TEIRESIAS (Davis, 1979)

Rule Editors

- **Knowledge acquired**
  - » **Parameters**
  - » **Conditional relations**
- **User knowledge required**
  - » **How to build problem-solving method from rules**
- **Support**
  - » **Guarantee legal syntax**
  - » **Provide help on what is possible in a given syntactic context**
  - » **Can be custom tailored to domains and special syntax**

Rule Editors (2)

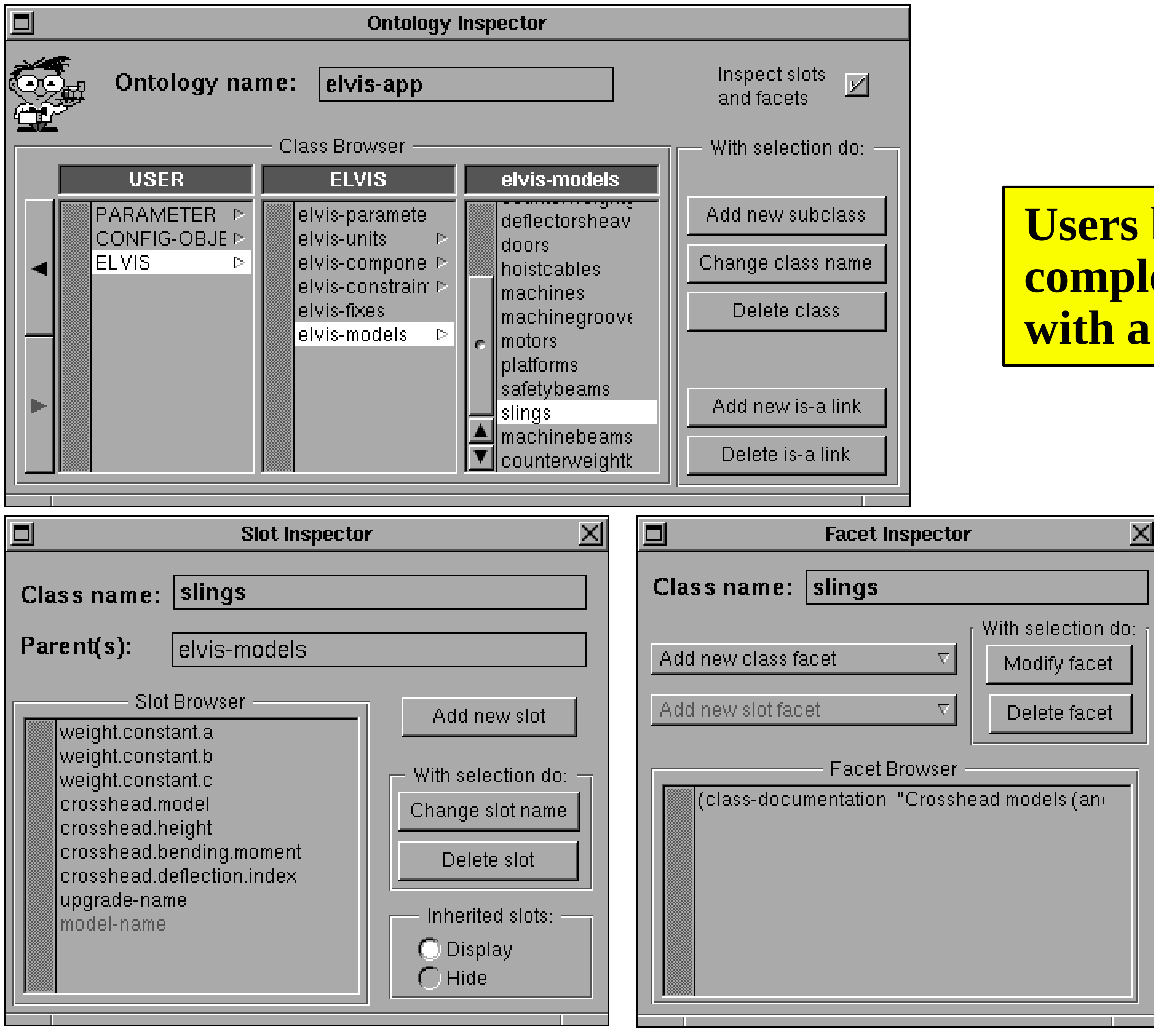
NEXPERT Object



Frame Editors

- **Knowledge acquired**
  - » **Concepts**
  - » **Relations**
- **User knowledge required**
  - » **How to build a model from frames and relations**
- **Examples**
  - » **MAÎTRE (Gennari, 1993)**
  - » **CODE (Skuce, 1993)**

Frame Editor Example: MAÎTRE

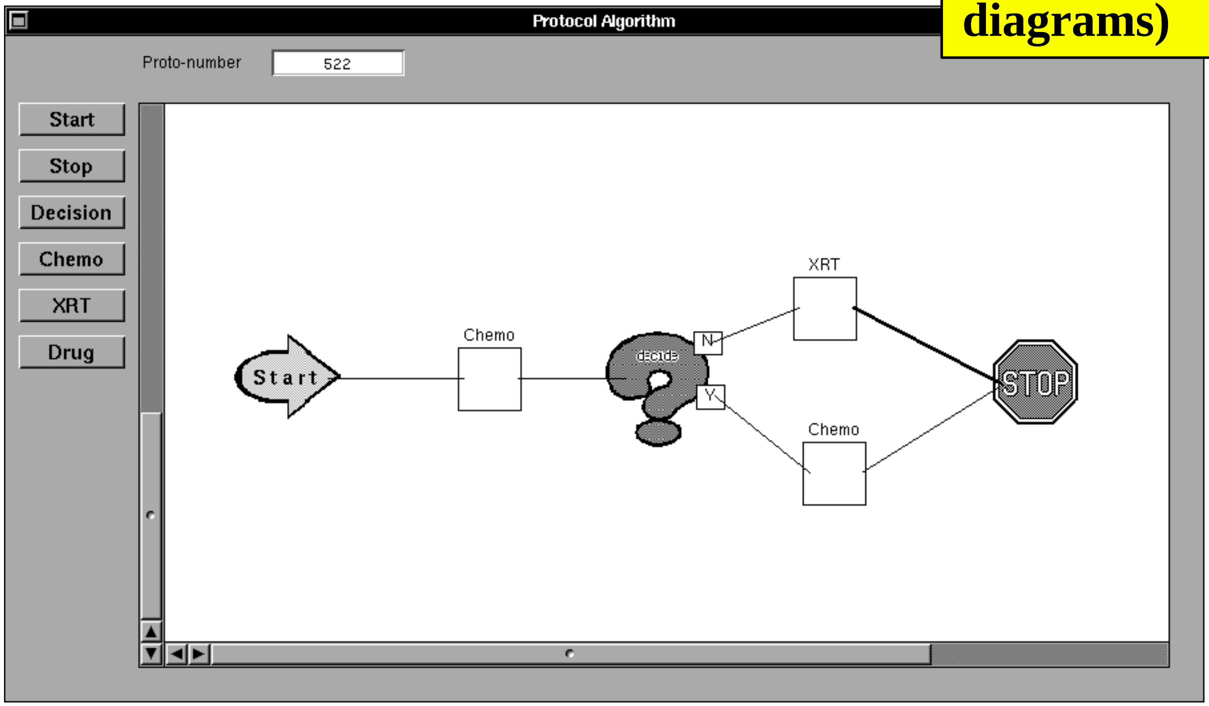


Users build and inspect complex class hierarchies with a browser-type editor

(Gennari, 1993)

Diagram Editors

Editors to acquire, inspect and review diagrammatic knowledge (e.g., influence diagrams)



TEIRESIAS: Rule Debugging

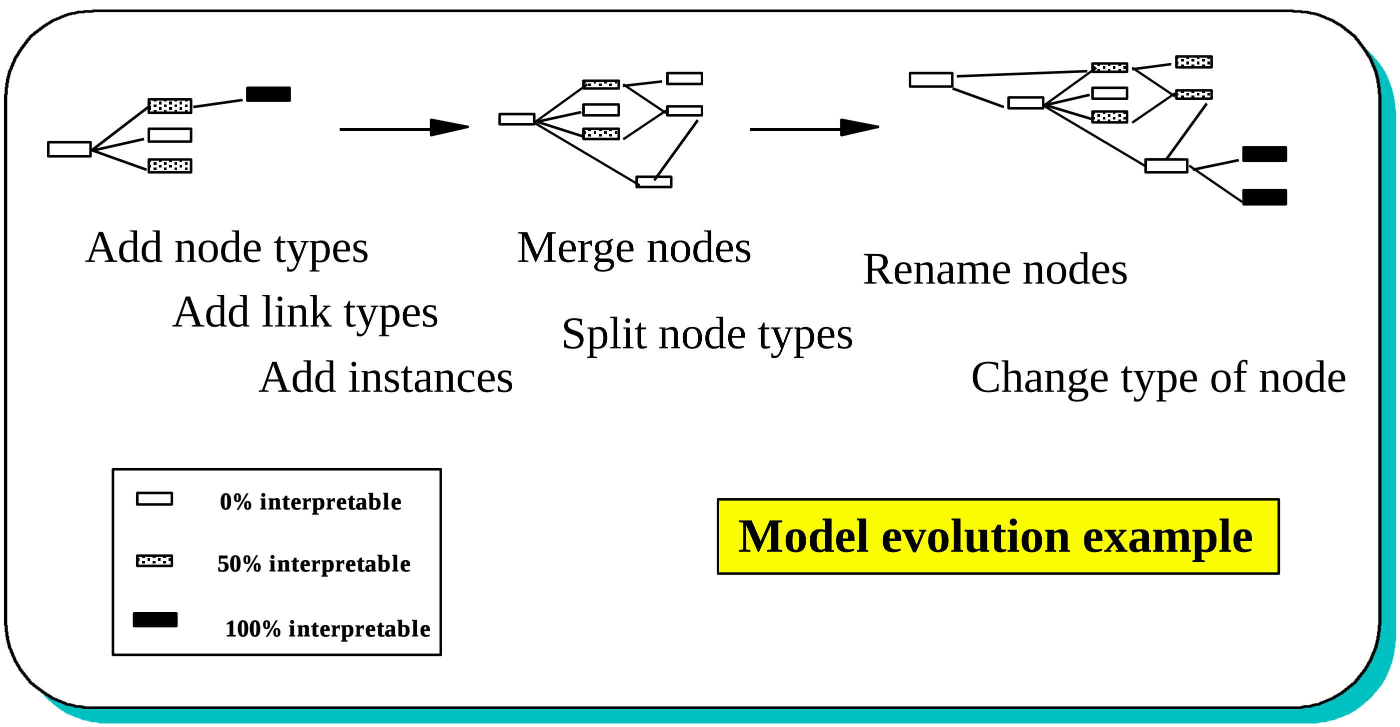
- Smart rule editor for MYCIN
- Committed to EMYCIN rule interpreter and representation language
- Intended for refinement of established knowledge systems
- Provides help with identifying and fixing “bugs” in the rule base
- Knowledge acquired: new and fixed rules
- User knowledge required: expertise in backward chaining, context trees, rule representation, domain knowledge

KA Model-Formulation Tools

- Tools that support definition of objects and relationships in models
- Resulting models are not intended to be executable
- Examples
  - » MeMo-Kit (Neubert and Mauer, 1993)
  - » Hypermedia interfaces
  - » Shelley (Anjewierden et al., 1990)
  - » Kibitzer (Schoen, 1990)

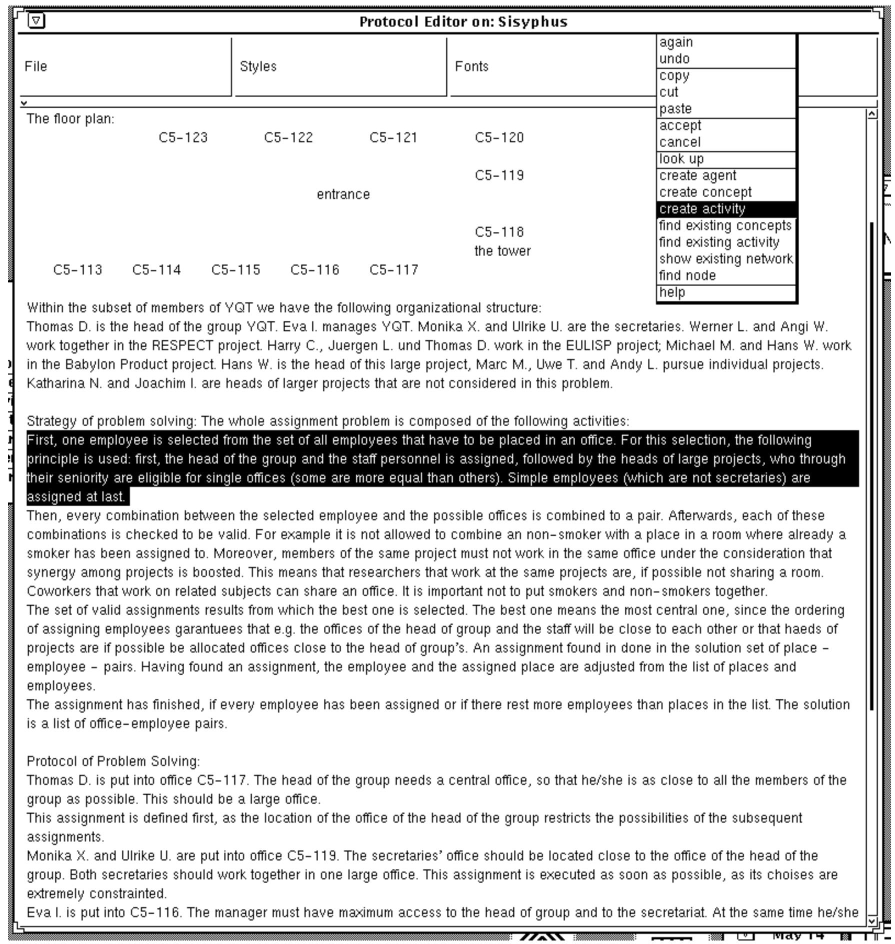
Hypermedia Interface

- **Class of tools to define semiformal representations**
  - » **Node and link *types***
  - » **Node and link *instances***
  - » **Node types: text, graphics, video, sound, etc.**



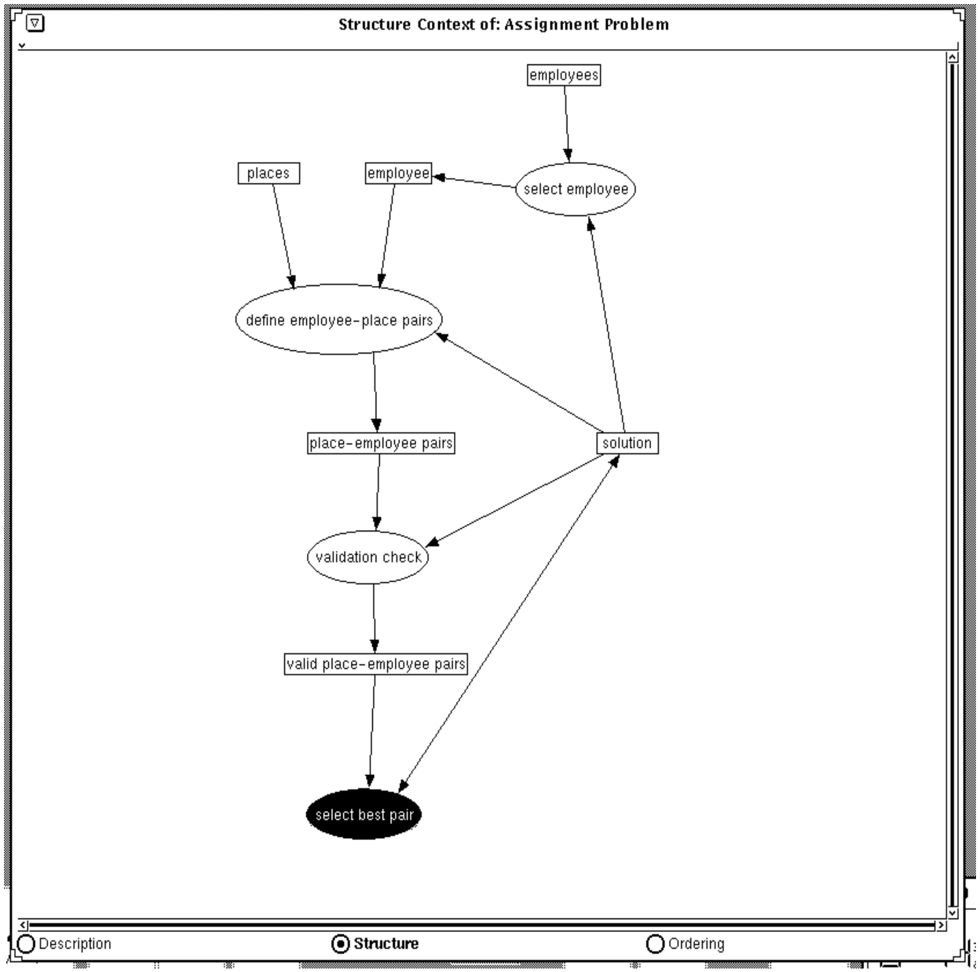
MeMo-Kit: Protocol Analysis

Editing knowledge at the domain theory level



MeMo-Kit: Activity and Concept Nodes

Editing knowledge at the operational level



Shelley: A KADS Workbench

- **A suite of tools to build KADS models**
- **Features:**
  - » **Lexical analysis of verbal protocols**
  - » **Graphing of conceptual relationships**
  - » **On-line access to textual KADS interpretation models**
  - » **Repertory-grid editor**

Kibitzer: E–R Modeling

- A tool to construct entity–relationship models
- Naming conventions
  - – Example: adjective stringing
- Properties of relations
  - » Example: transitivity, invertability
- Style heuristics
  - – Example: “bushiness” of taxonomies
- Mapping of entities and relations to target representations

KA Method-Specific Tools

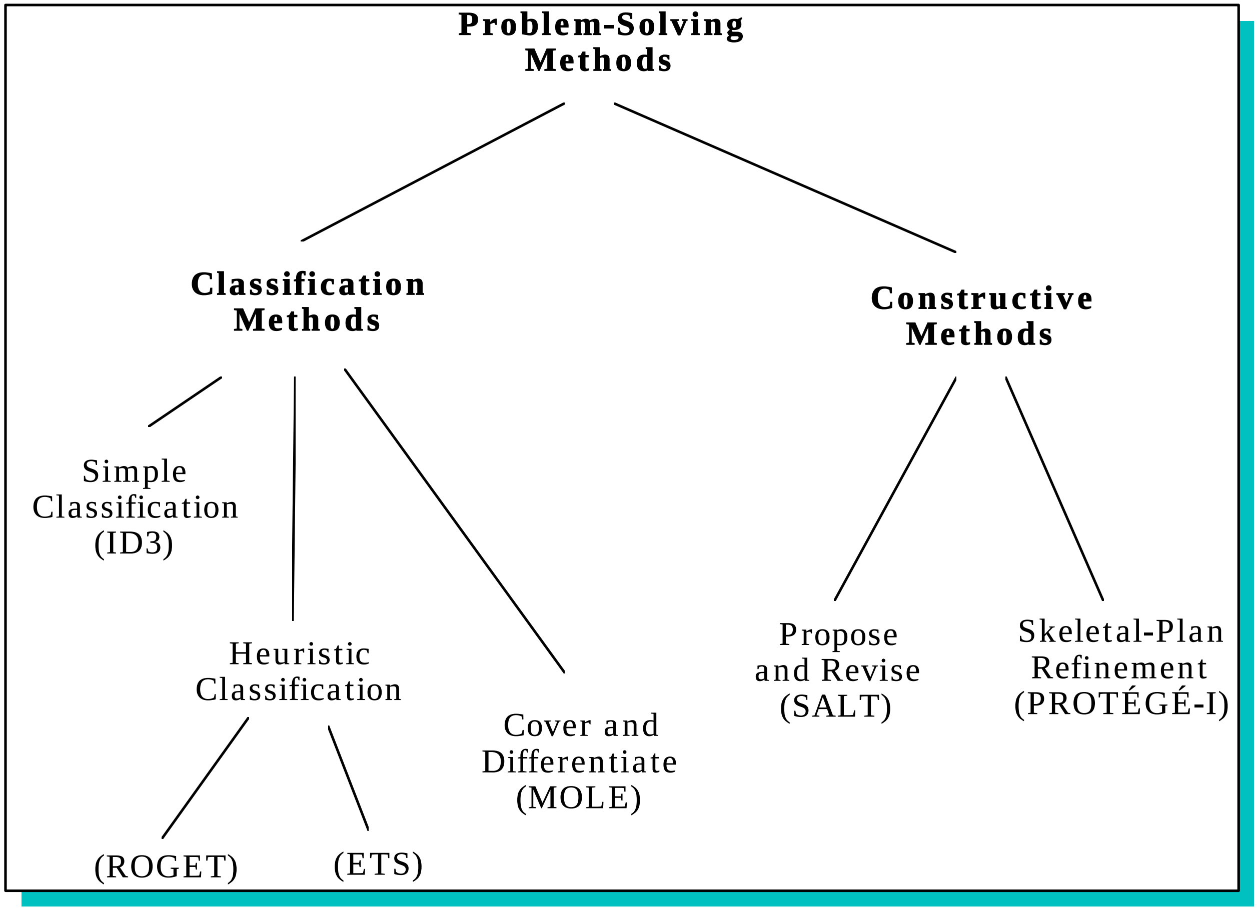
Tools that make assumptions about problem-solving methods

- **Make commitment to**
  - » **the representation language of the performance system**
  - » **a predefined problem-solving method**
- **Higher level of abstraction than symbol-level tools**
- **Examples:**

KA Tool	Problem-Solving Method	Reference
ETS, AQUINAS	Hierarchical Classification	Boose 1985; Boose and Bradshaw 1987
SALT	Propose and Revise	Marcus 1987; Marcus and McDermott, 1989
ROGET	Heuristic Classification	Bennet 1985

KA Method-Specific Tools (2)

Partial classification of tools according to their underlying method



SALT: Constructive Problem Solving

- **Knowledge-acquisition tool for the VT system (elevator configuration) (Marcus et al., 1988)**
- **Supports the *propose-and-revise* method**
- **Knowledge acquired**
  - » **Design extensions**
  - » **Constraint checking**
  - » **Backtracking from constraint violations**
- **User knowledge required**
  - » **Relating task features to procedures, constraints, and fixes**

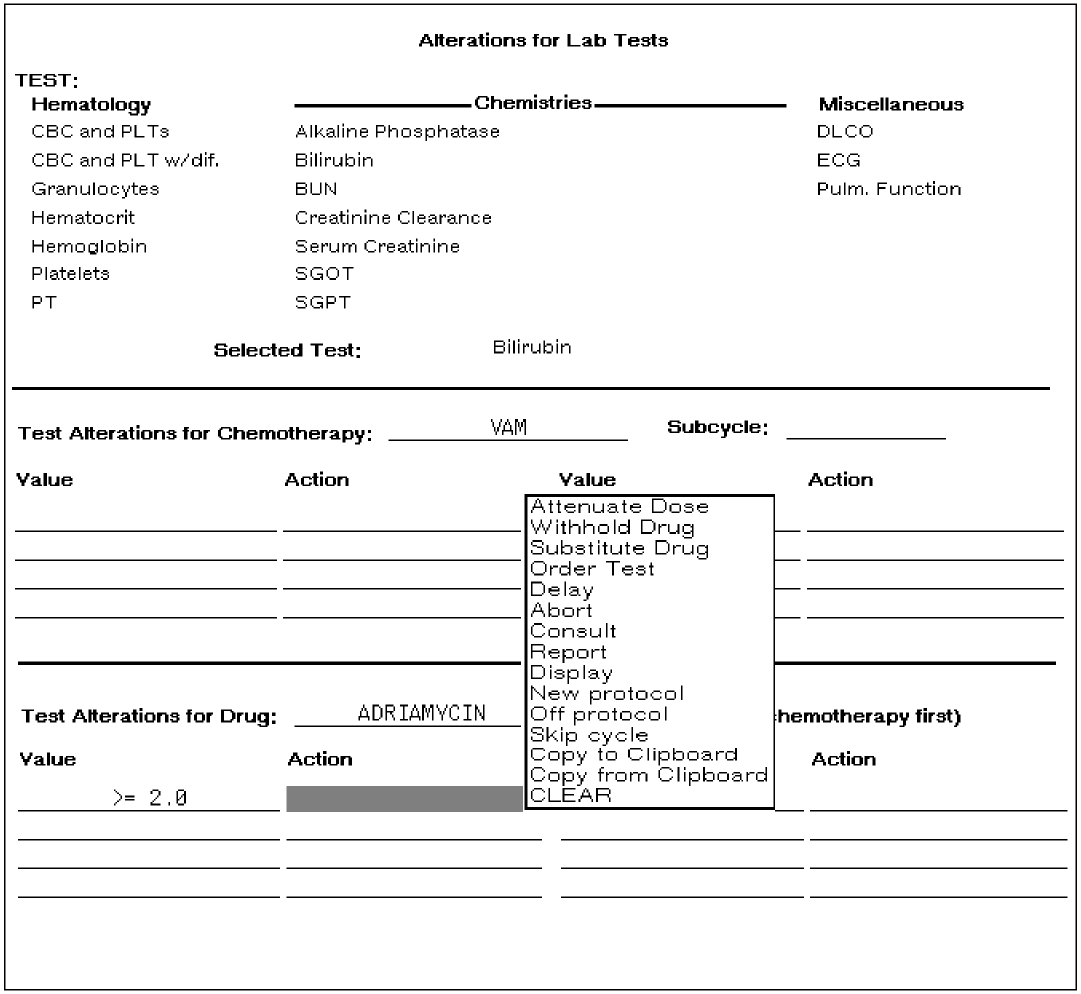
KA Domain-Specific Tools

- Tools that incorporate domain concepts
- Custom tailored for domain experts
- Relatively high tool-development cost
- Examples:
  - » OPAL (Planning of cancer therapy) (Musen, 1987)
  - » P10 (Planning of protein purification) (Eriksson, 1992a)

OPAL:
KA for Cancer-Therapy Administration

- **KA tool for ONCOCIN expert system (Tu et al., 1989)**
- **Knowledge acquired**
  - » **Experimental cancer-treatment plans (protocols)**
- **User knowledge required**
  - » **Cancer-therapy expertise**

Eliciting Details of Drug Administration in OPAL



Alterations for Lab Tests

TEST:

Hematology	Chemistries	Miscellaneous
CBC and PLTs	Alkaline Phosphatase	DLCO
CBC and PLT w/dif.	Bilirubin	ECG
Granulocytes	BUN	Pulm. Function
Hematocrit	Creatinine Clearance	
Hemoglobin	Serum Creatinine	
Platelets	SGOT	
PT	SGPT	

Selected Test: Bilirubin

Test Alterations for Chemotherapy: VAM Subcycle:

Value Action Value Action

Attenuate Dose
Withhold Drug
Substitute Drug
Order Test
Delay
Abort
Consult
Report
Display
New protocol
Off protocol
Skip cycle
Copy to Clipboard
Copy from Clipboard
CLEAR

Test Alterations for Drug: ADRIAMYCIN …hemotherapy first)

Value Action Action

>= 2.0

OPAL provides domain-specific forms for knowledge elicitation

Knowledge Maintenance and Processing

- **Most maintenance and processing is done via *expert-system shells***
- **An expert-system shell provides:**
  - » **Rudimentary facilities for knowledge editing**
  - » **A reasoning engine**
  - » **Consistency-checking facilities**
- **Examples:**
  - » **ART (Rule based)**
  - » **Knowledge craft (Rule based)**
  - » **GBB (Blackboard based)**
  - » **Clips (Frame based)**

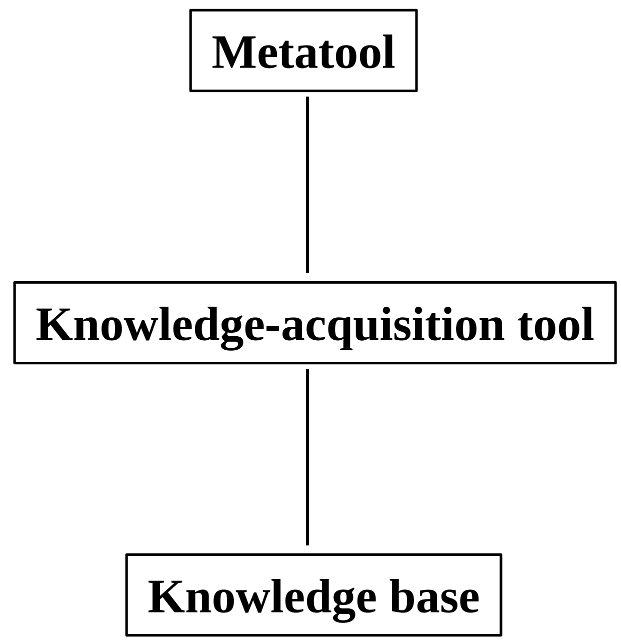
Toward Knowledge Environments

- **It can be observed from the tool review that:**
  - » **Knowledge engineering requires working with expert-system shells and KA tools**
  - » **Building an expert system implies building a KA tool!**
  - » **No KE tool shown provides support for all facets of expert system development**
  - » **There is a need for comprehensive software environments for knowledge engineering**

A knowledge-engineering environment is an integrated suite of software tools that supports all facets of construction and use of expert systems and of knowledge-acquisition tools

Metatools for Knowledge Acquisition

- **Tools that *generate* knowledge-acquisition tools**
- **Provide a standard reasoning engine**
- **Two-layer approach**
- **Examples:**
  - » **PROTÉGÉ-I (Musen, 1989a, b)**
  - » **DOTS (Eriksson, 1992b, 1993)**
  - » **SIS (Kawaguchi et al., 1991)**



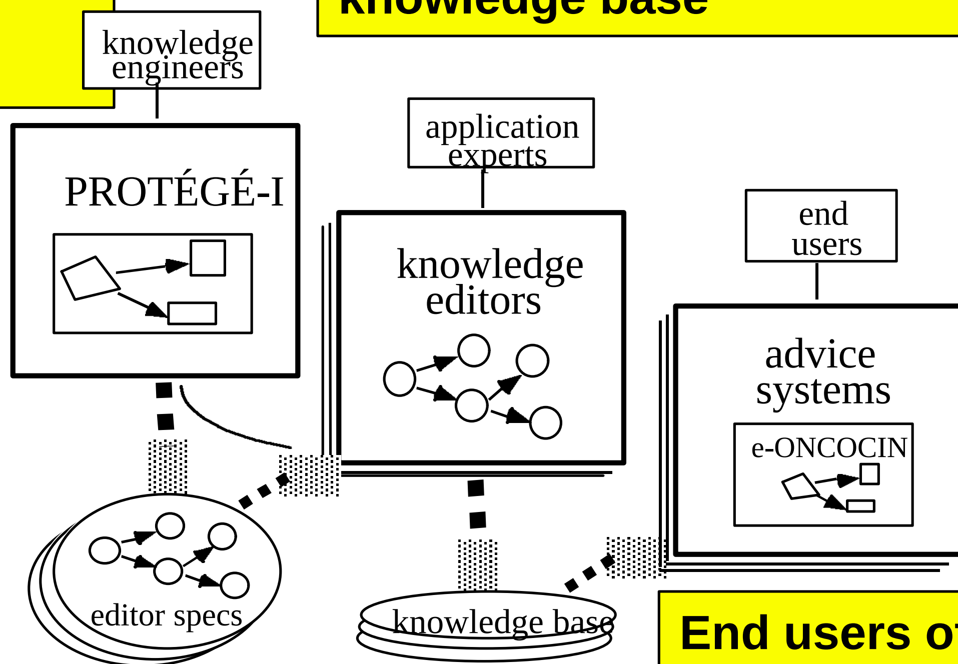
PROTÉGÉ-I: Generating OPAL-Class Tools

- **Assumes skeletal-plan–refinement method.**
- **Generates task- and domain-specific KA tools**
- **Separates knowledge acquisition into two phases:**
  - » **Knowledge-level analysis (task modeling)**
  - » **Entry of content knowledge**

The PROTÉGÉ-I Approach

Knowledge engineers generate knowledge editors by instantiation of the skeletal-plan method

Domain experts complete knowledge base



End users of expert systems utilize problem-solving method and knowledge base

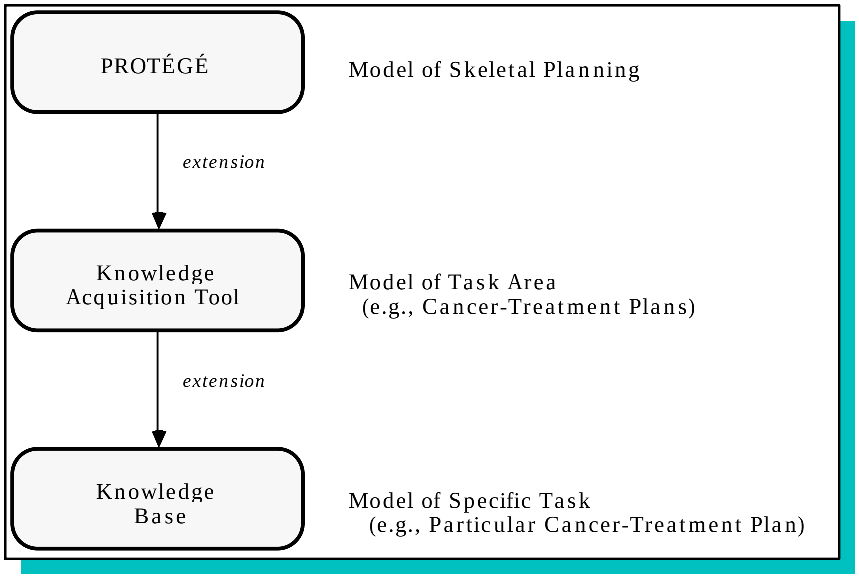
PROTÉGÉ-I Summary

Knowledge acquired

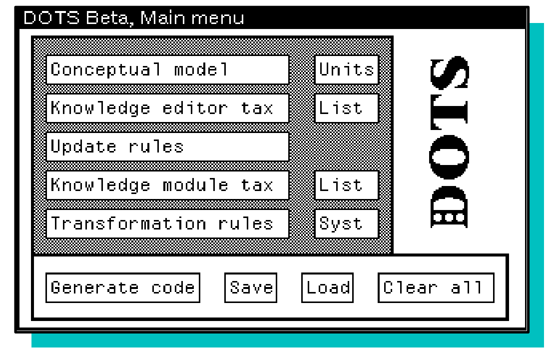
- » **Domain-specific versions of planning concepts**

User knowledge required

- » **Relating features of domain to planning concepts**

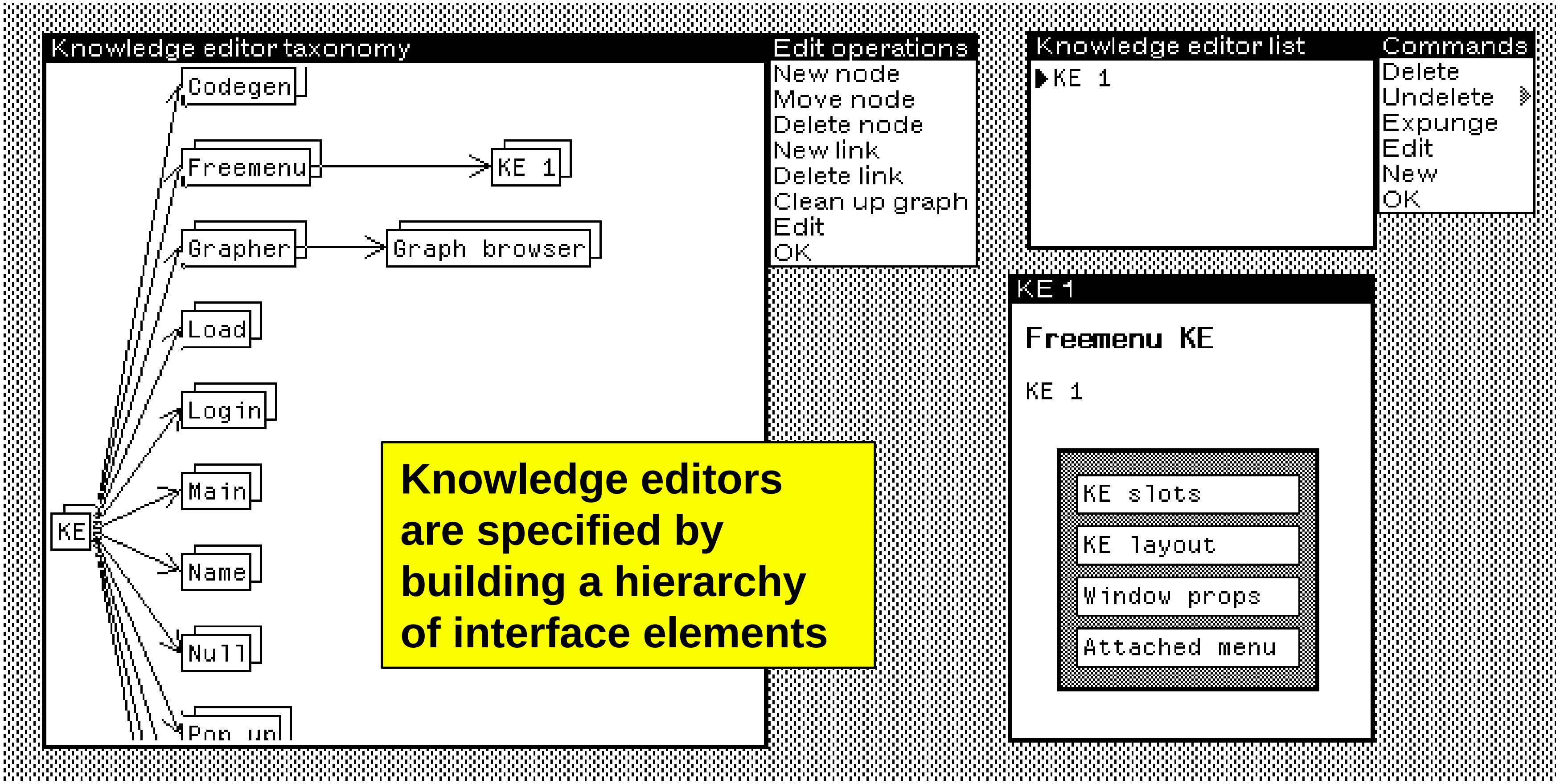


DOTS:
Generating P10-Class Tools



- **Domain-Oriented Tool Support**
- **Method-independent metatool**
- **Makes assumption about target tools**
- **Specification of KA tools in terms of**
  - » ***knowledge editors* for graphical knowledge entry**
  - » ***knowledge modules* for knowledge representation in the target tool**
  - » ***update rules* for communication among knowledge editors and modules**
  - » ***transformation rules* for generation of knowledge bases**

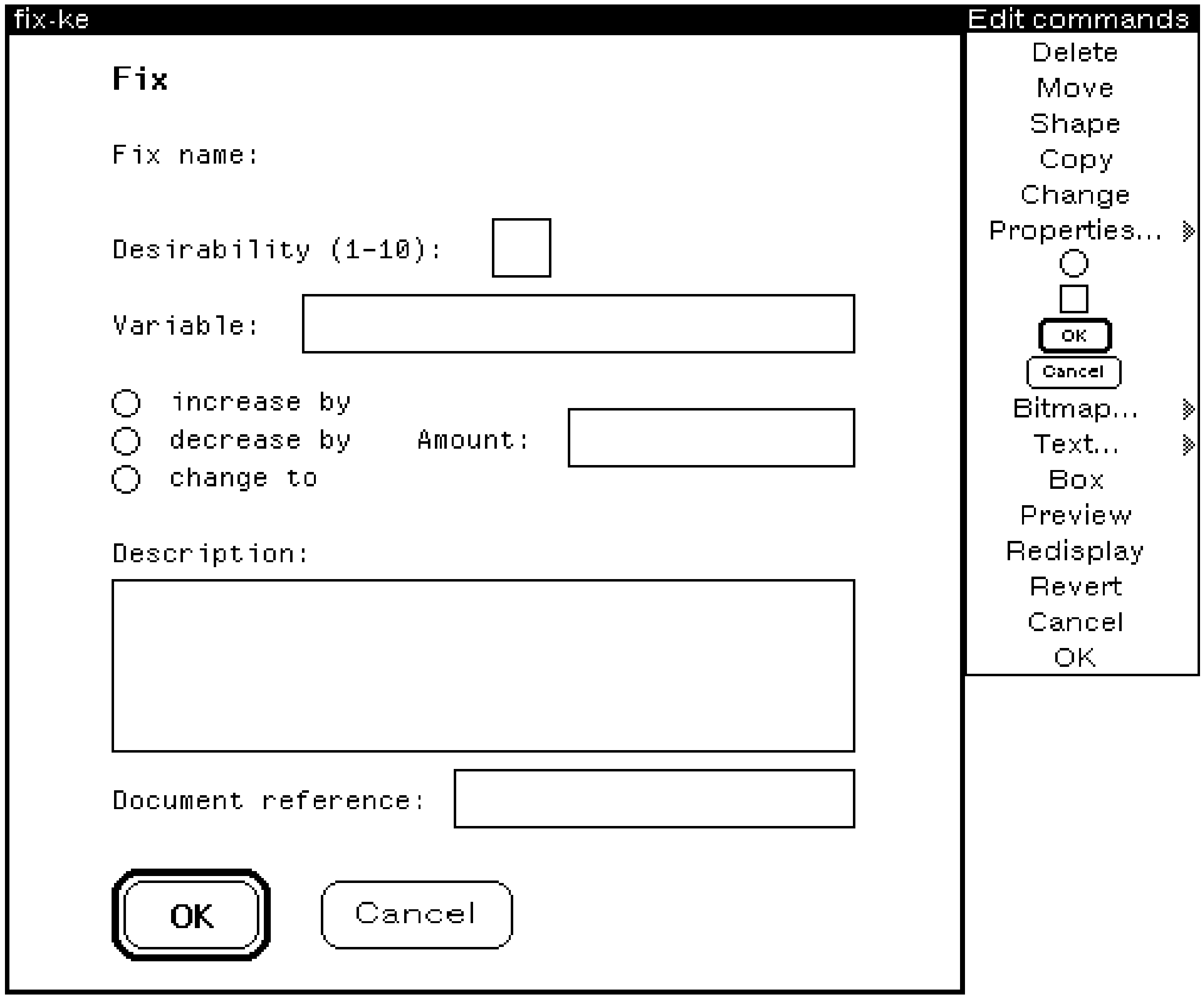
DOTS:
Specification of Knowledge Editors



Knowledge editors are specified by building a hierarchy of interface elements

Layout Design in DOTS

The layout of knowledge editors is specified via dialog panels



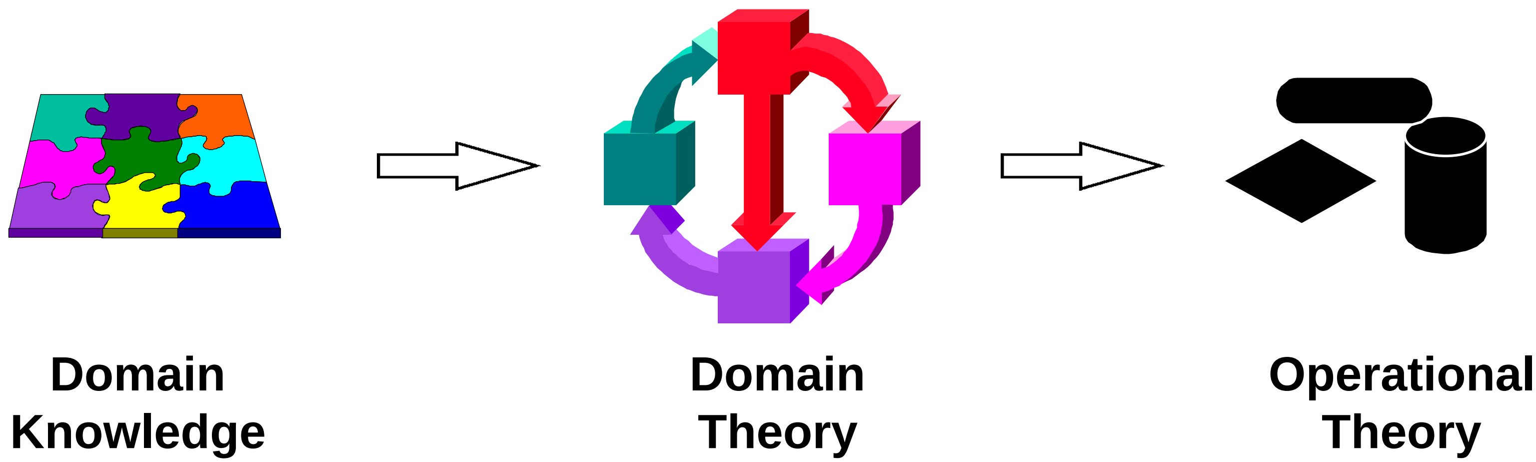
DOTS Summary

- Architectural view of knowledge-acquisition tools
- DOTS is more general than PROTÉGÉ-I, but requires more manual design work than does PROTÉGÉ-I
- Bootstrapped implementation
- KA tools developed using DOTS:
  - » Troubleshooting of laboratory equipment (DNA sequencing machines)
  - » Sisyphus room-assignment task
  - » Sisyphus VT task (elevator configuration)

More Shortcomings of Current Tools

- **Costly design and development**
  - » **Many KA tools designed for a specific expert system**
  - » **Many KA tools developed *after* its target expert system**
- **Difficult to reuse KA tools for other applications**
- **Poor life cycle support**
  - » **No support for method selection, debugging, and maintenance**
- **Limited support for knowledge and software reuse**
  - » **No support for development of expert systems from reusable components**

Knowledge-Engineering Environments for Reusable Components

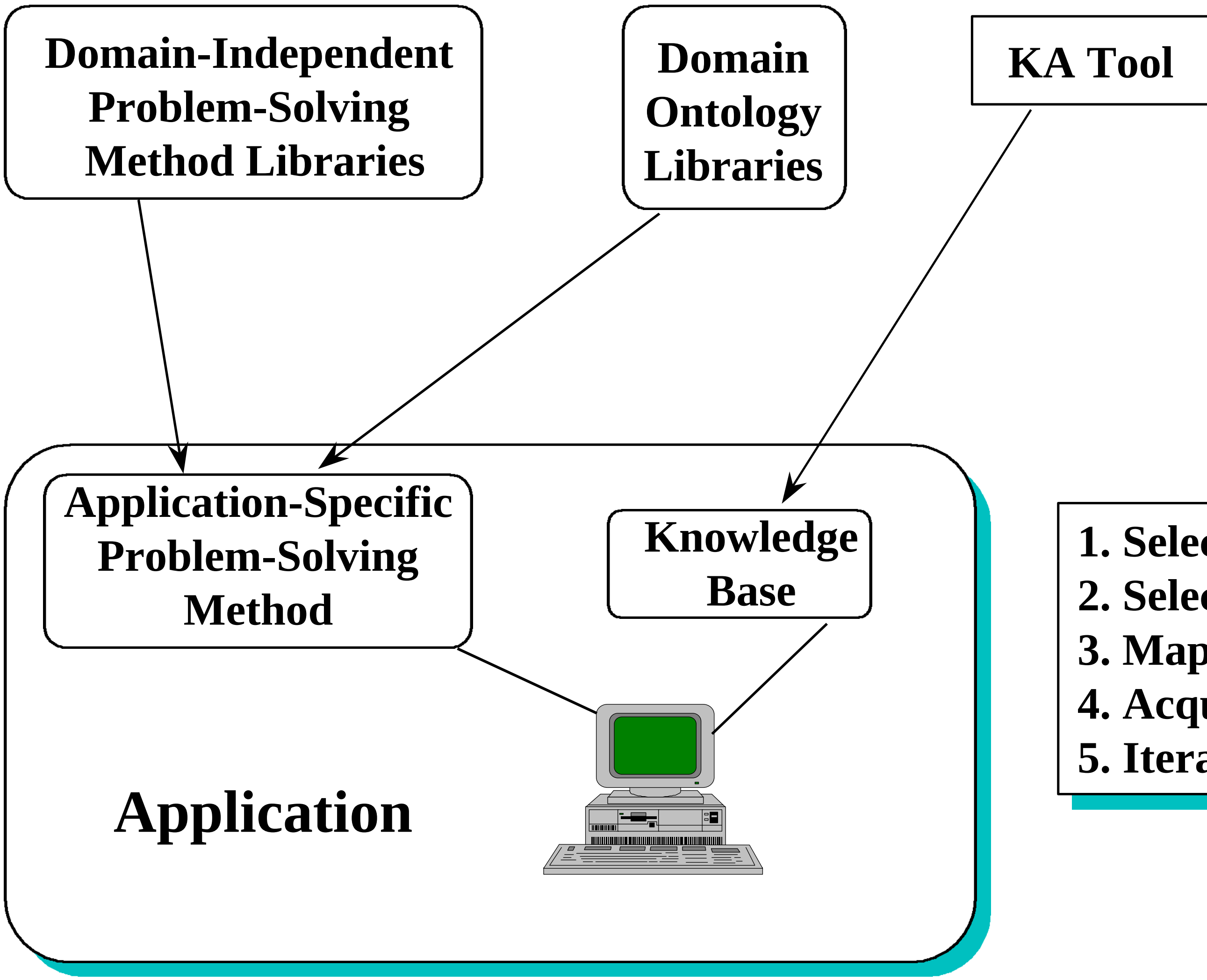


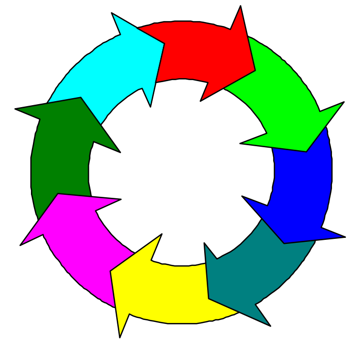
- **Integrated suites of tools that:**
  - » Support development cycles based on reuse
  - » Provide access to libraries of problem-solving methods
  - » Provide access to libraries of domain ontologies
  - » Support knowledge-acquisition metatools
  - » Support several user categories (e.g., developers, experts)

Reuse in Knowledge-Based Systems Development

- **Researchers and developers have found that:**
  - » **Many systems apply similar problem-solving strategies**
  - » **There are patterns and classes of problem-solving strategies**
  - » **Domains can be represented explicitly and stored**
- **Developers need environments that:**
  - » **Provide libraries of problem-solving methods (inference engines)**
  - » **Provide libraries of domain ontologies**
  - » **Support selection and review of methods and domain ontologies**
  - » **Support knowledge acquisition**
  - » **Support the entire knowledge-based system development cycle**

A Development Cycle for Reuse





1. Select/Edit method
2. Select/Edit domain ontology
3. Map method to domain
4. Acquire application knowledge
5. Iterate

Internet-based approaches to knowledge engineering

Internet Implications for:

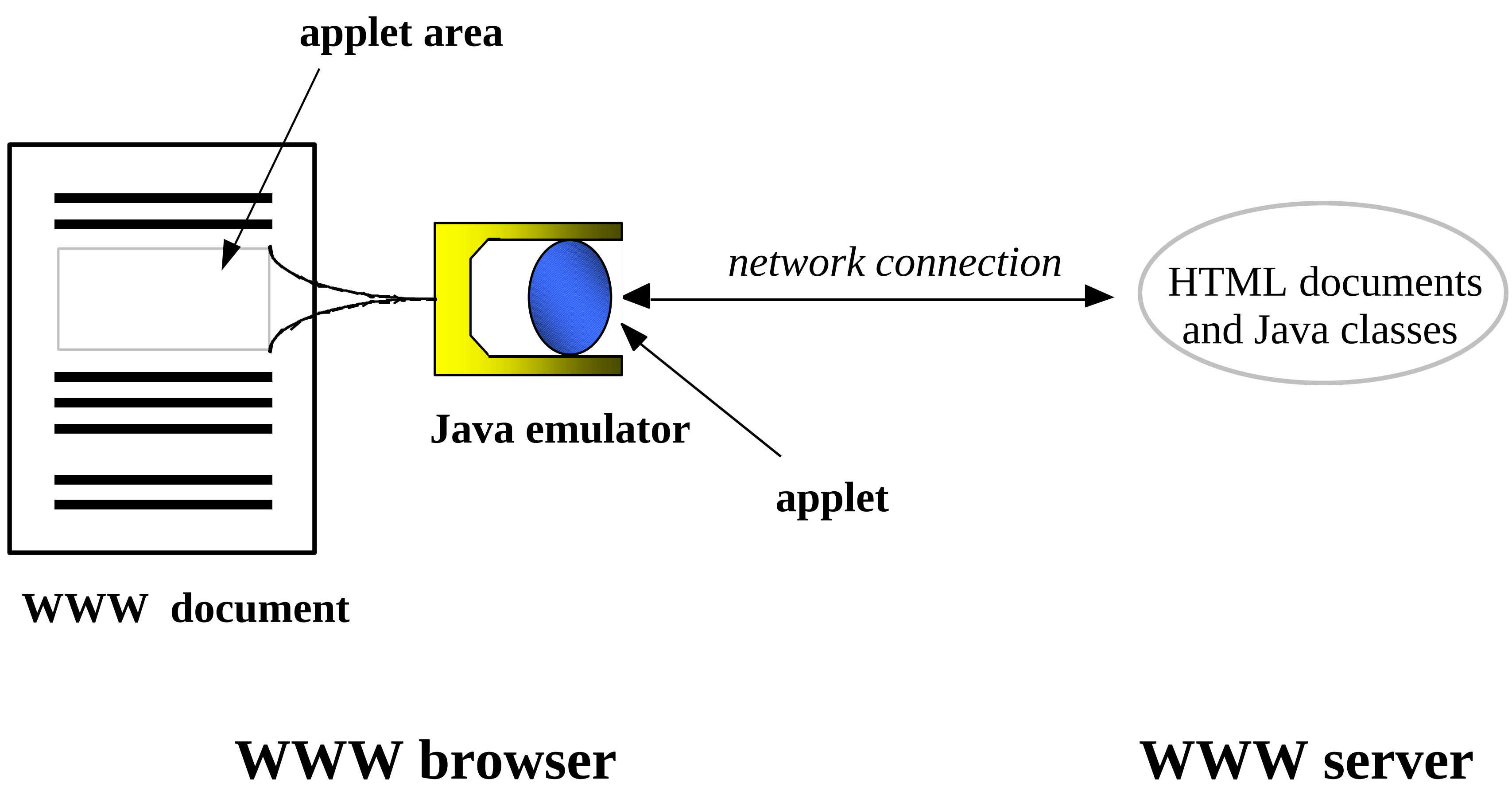
- **Architectures**
  - » **New distributed architectures for knowledge-based systems**
  - » **New classes of target systems**
- **Development assistance**
  - » **Cooperative design work**
  - » **Network-based development tools**
  - » **Distributed design information (e.g., vocabularies, terminology servers)**

Relevant Internet Technologies

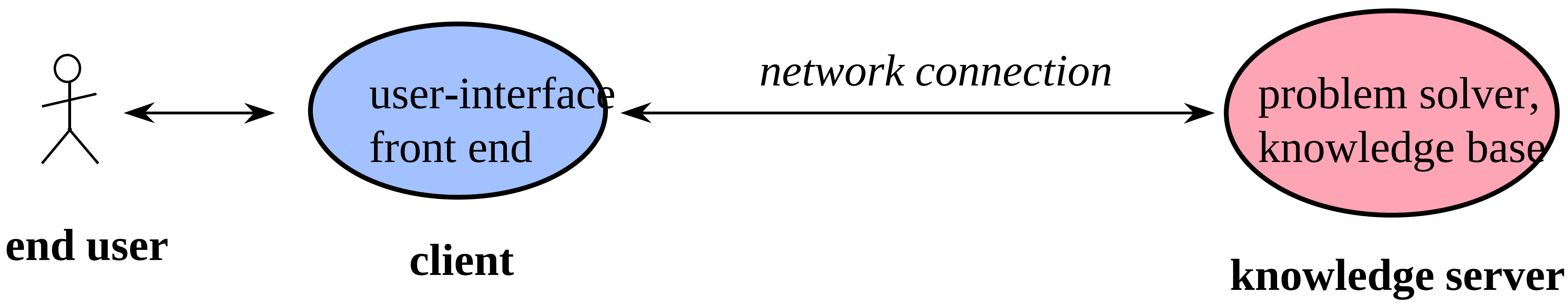
- **E-mail**
- **World-Wide Web (WWW)**
- **Java and Java applets**
- **Jess: CLIPS emulation in Java**
- **CORBA**
- **Terminology servers**
- **Computer-supported cooperative work (CSCW) solutions**

Knowledge-based systems can benefit from Java technology

Java Applets (summary)



Two-Component Architecture

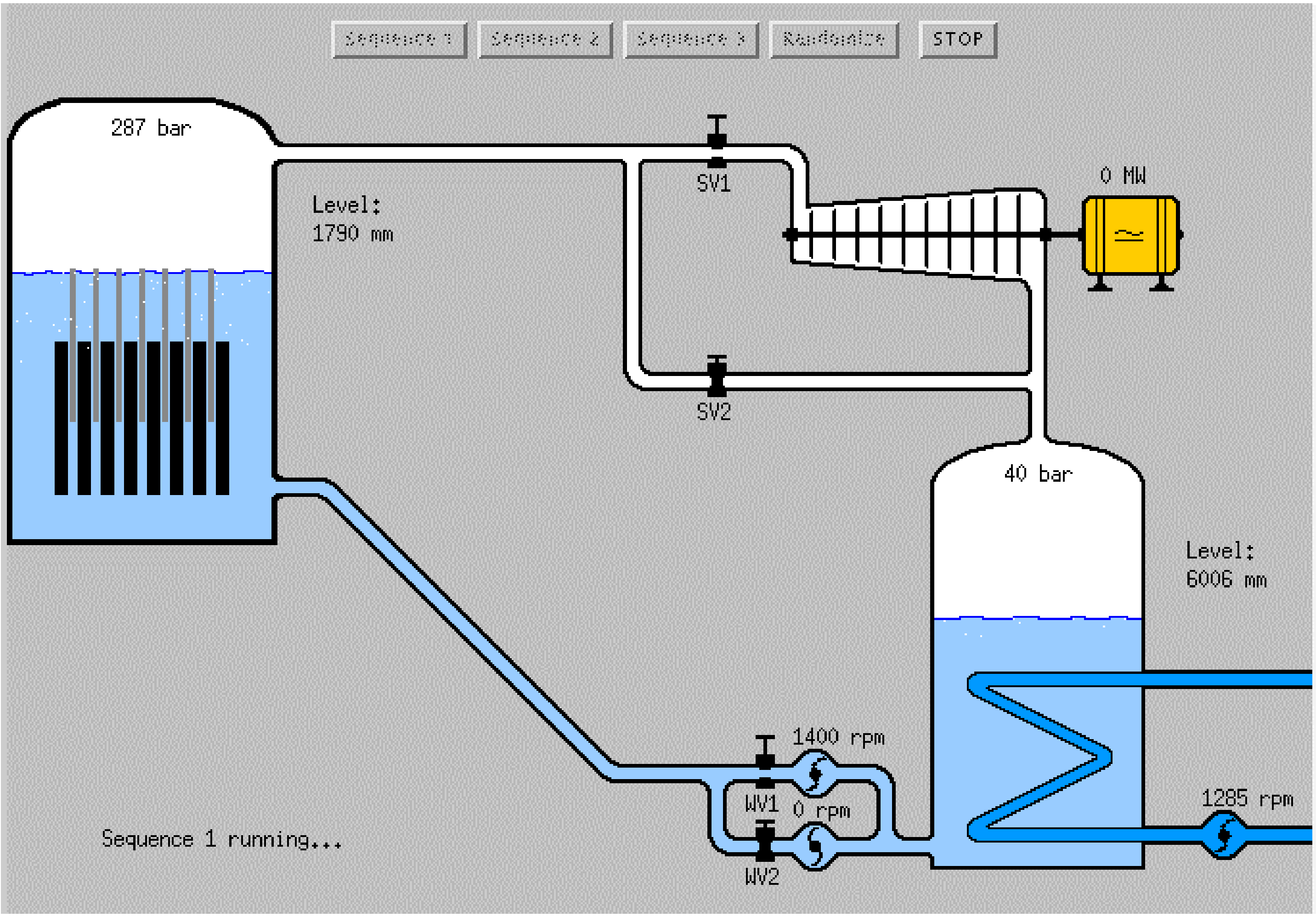


Knowledge-based systems can benefit from client–server solutions

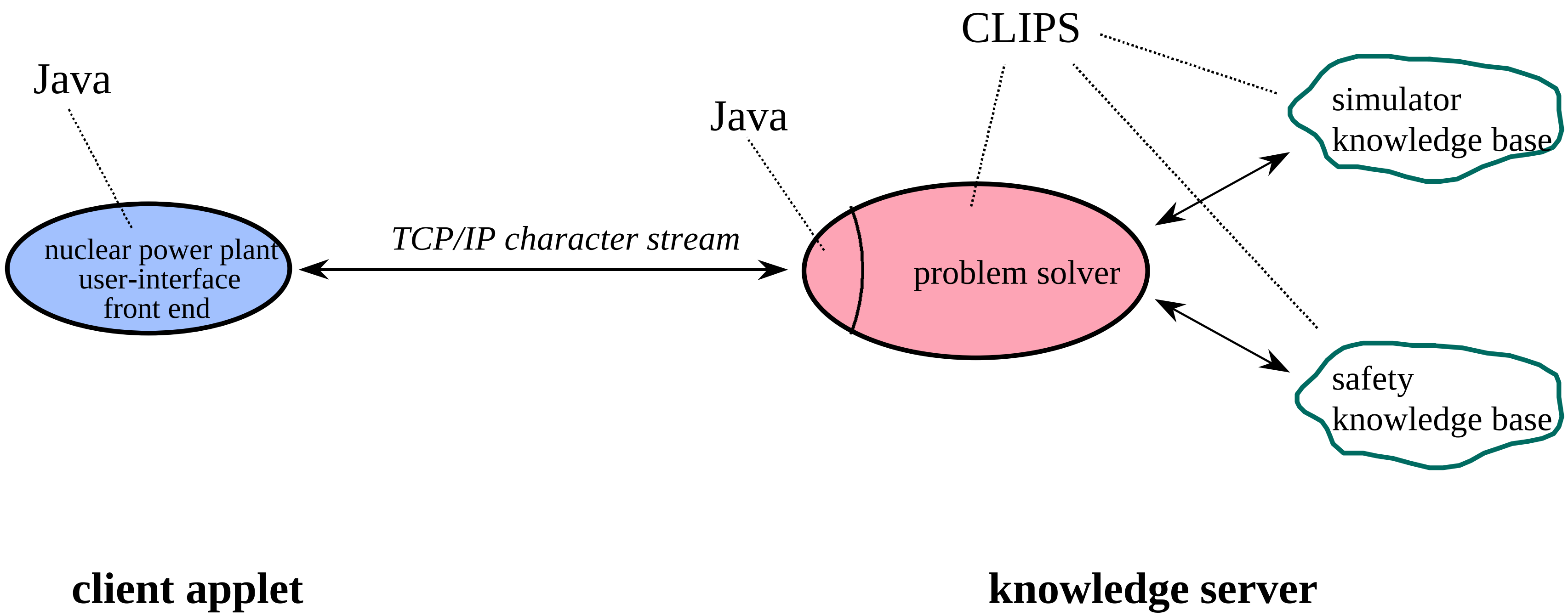
Sample Client–Server System

- **Basic simulation of nuclear power plants: Chernobyl**
- **Task: Avoid further damage when components fail**
- **Programming assignment for undergraduates**
  - » **Learn to control the power plant manually**
  - » **Implement rules for automatic control**
- **Client–server architecture**
  - » **User interface in Java**
  - » **Server in Java and CLIPS**
- **Undergraduates create their own knowledge server**
- **Location: http://www.ida.liu.se/~her/npp/demo.html**

Chernobyl Applet



Kärnobyl Architecture



Undergraduates implement the safety knowledge base only

Internet-based Environments

- Ontolingua WWW server
  - » On-line ontology editor
- Sun Microsystems' Java WorkShop
  - » Implemented in Java
  - » Based on the HotJava WWW browser
- Repertory-grid tools
  - » GCI interface
- Wanted: On-line libraries of reusable problem-solving methods

Summary: Current Trends in KE Tools

Expert system development requires use of sophisticated ES shells and system-specific KA tools

Trends
- » Comprehensive software environments
- » Reuse of knowledge
- » Internet technologies

The next step: KE environments for reusable components on the Internet

Agenda

- Knowledge engineering concepts
- Current trends in knowledge-based development
- Break
- Case Studies
- Incorporating knowledge engineering tools into software projects
- Summary: Lessons learned and future directions
- Questions

Agenda

- Knowledge engineering concepts
- Current trends in knowledge-based development
- Break
- <u>Case Studies</u>
- Incorporating knowledge engineering tools into software projects
- Summary: Lessons learned and future directions
- Questions

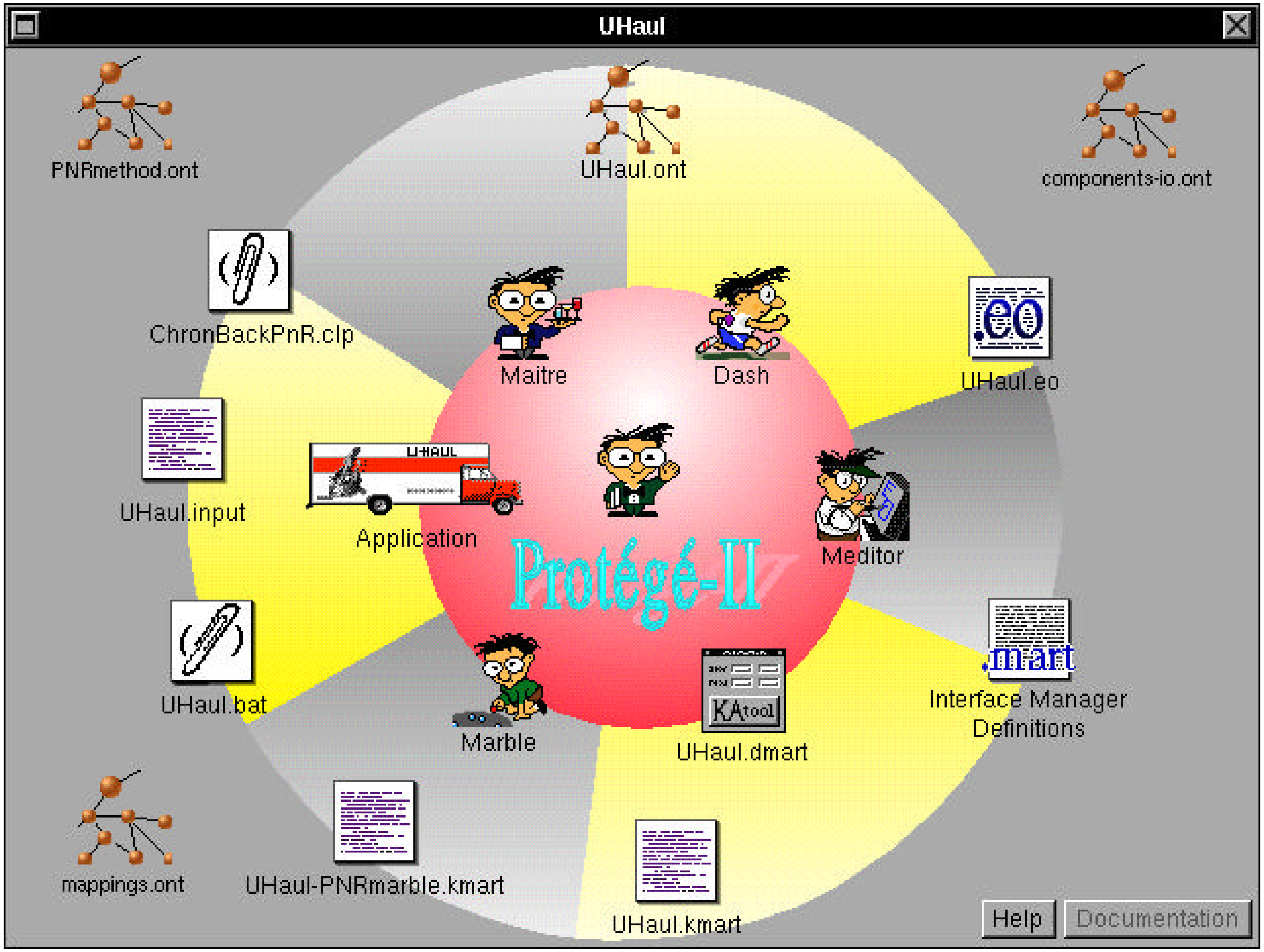
KE Environments: Case Studies

- **Objectives:**
  - » **Illustrate use of KE environments**
  - » **Understand different technical approaches**
  - » **Identify benefits and shortcomings of each system**
- **Method-oriented architectures**
  - » **PROTÉGÉ-II (Puerta et al., 1992)**
  - » **DIDS (Runkel and Birmingham, 1993)**
- **Non-programmers**
  - » **SBF (Klinker et al., 1991; Marques et al., 1992)**
- **Knowledge-level and KADS-oriented systems**
  - » **KREST (Steels, 1993)**
  - » **VITAL (Shadbolt et al., 1993)**

PROTÉGÉ-II

- A KE environment for development of knowledge-based systems
- Emphasizes reuse of problem-solving methods
- Generation of knowledge-acquisition tools from domain ontologies (DASH) (Eriksson et al., 1994)
- Runtime system for knowledge-acquisition tools (MART) (Puerta et al., 1994)

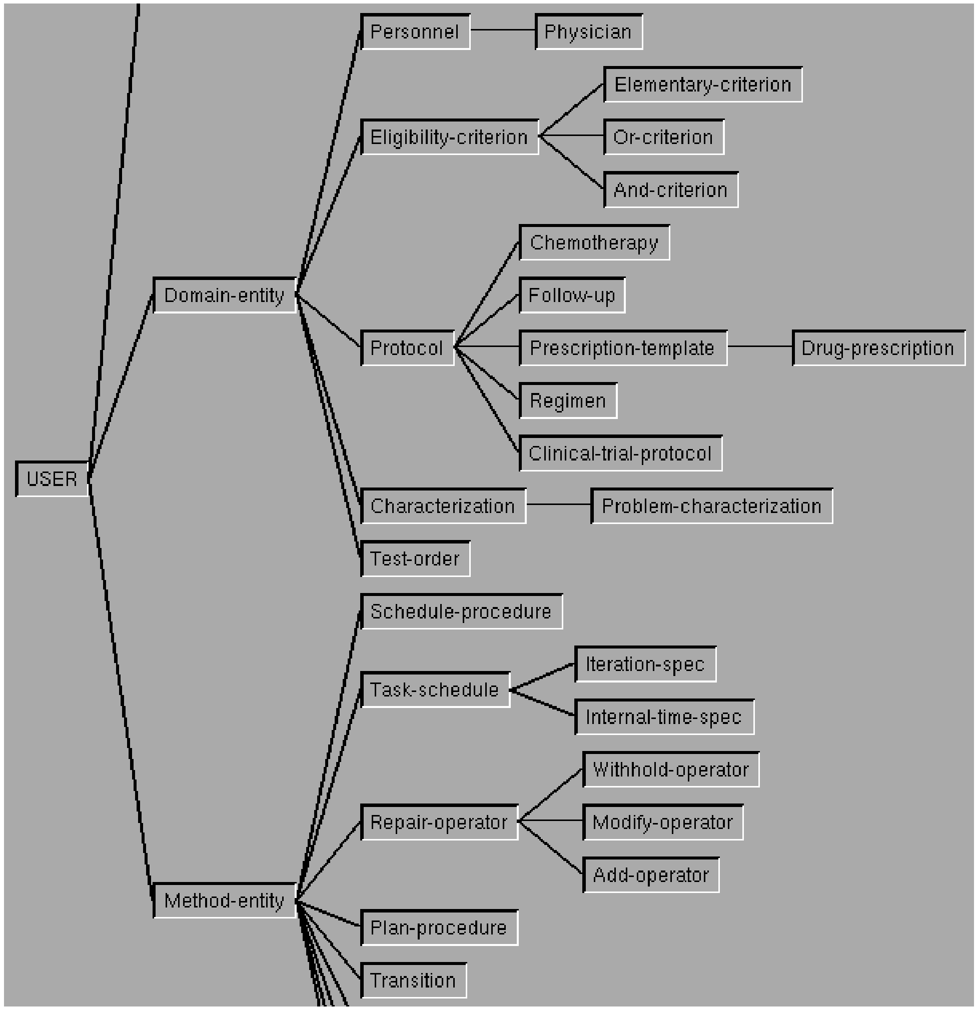
PROTÉGÉ-II (2)



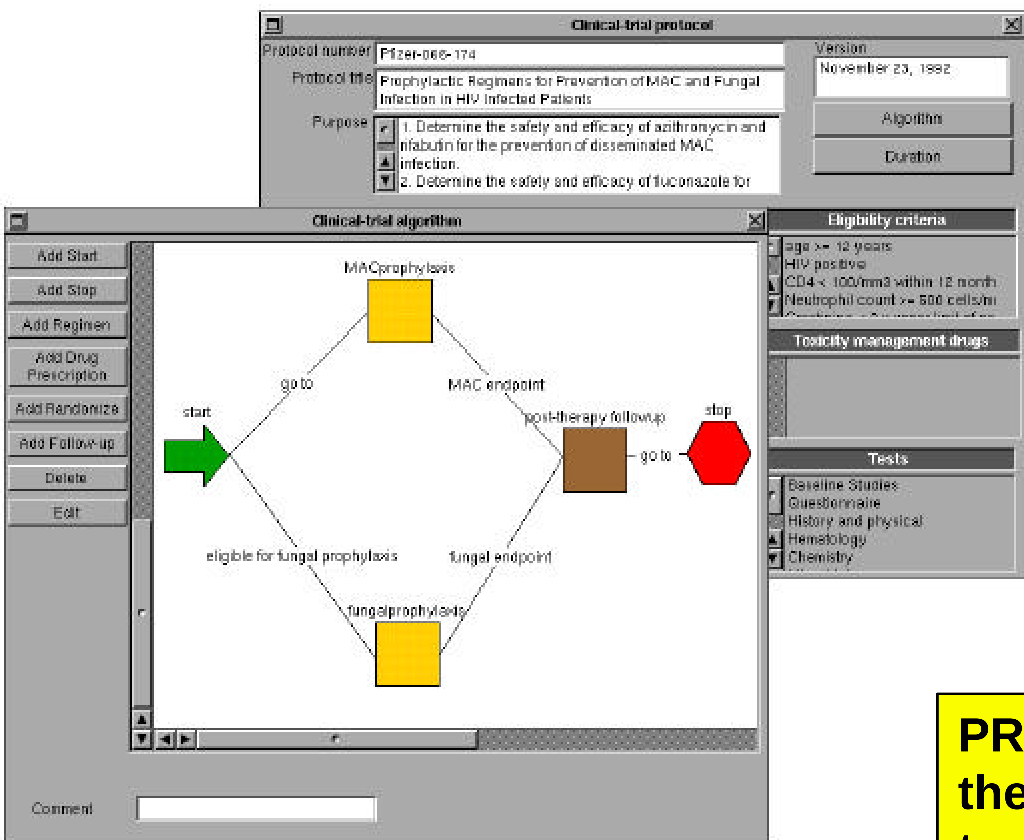
PROTÉGÉ-II delivers a development environment that emphasizes automation, reusability, and early prototyping

Example: Domain Ontology

Ontologies in PROTÉGÉ-II are defined as class hierarchies that can be inspected and edited through a browser-like ontology editing tool



Example: Knowledge-Acquisition Tool



PROTÉGÉ-II supports the generation of KA tools for use by domain experts

PROTÉGÉ-II: The DASH Metatool

- Generates knowledge-acquisition tools from domain ontologies
- Separates KA tool design into two levels of abstraction
  - » dialog design
  - » layout design
- Allows the developer to custom-tailor the target tool

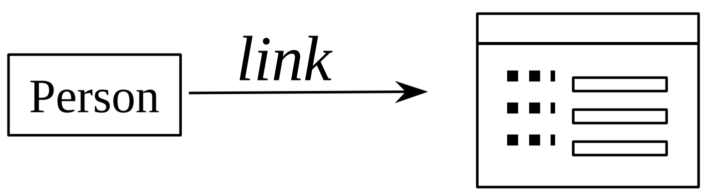
DASH: Ontologies as the Basis for KA Tool Generation

Basic ideas:

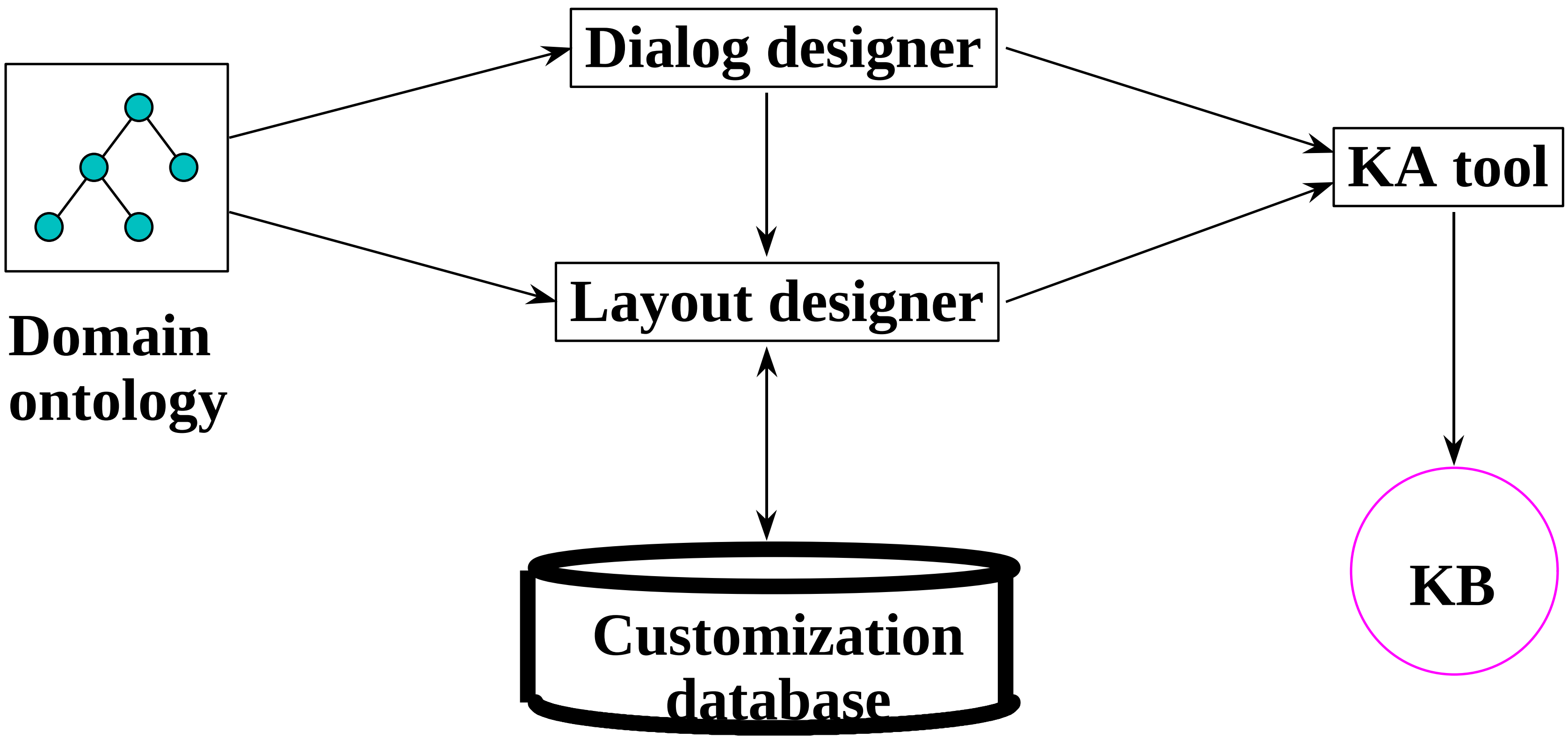
- Use data types from slot definitions to generate form fields
- Use relationships (links) among class definitions to generate the dialog structure

```
(slot chapter11 (type boolean))
```

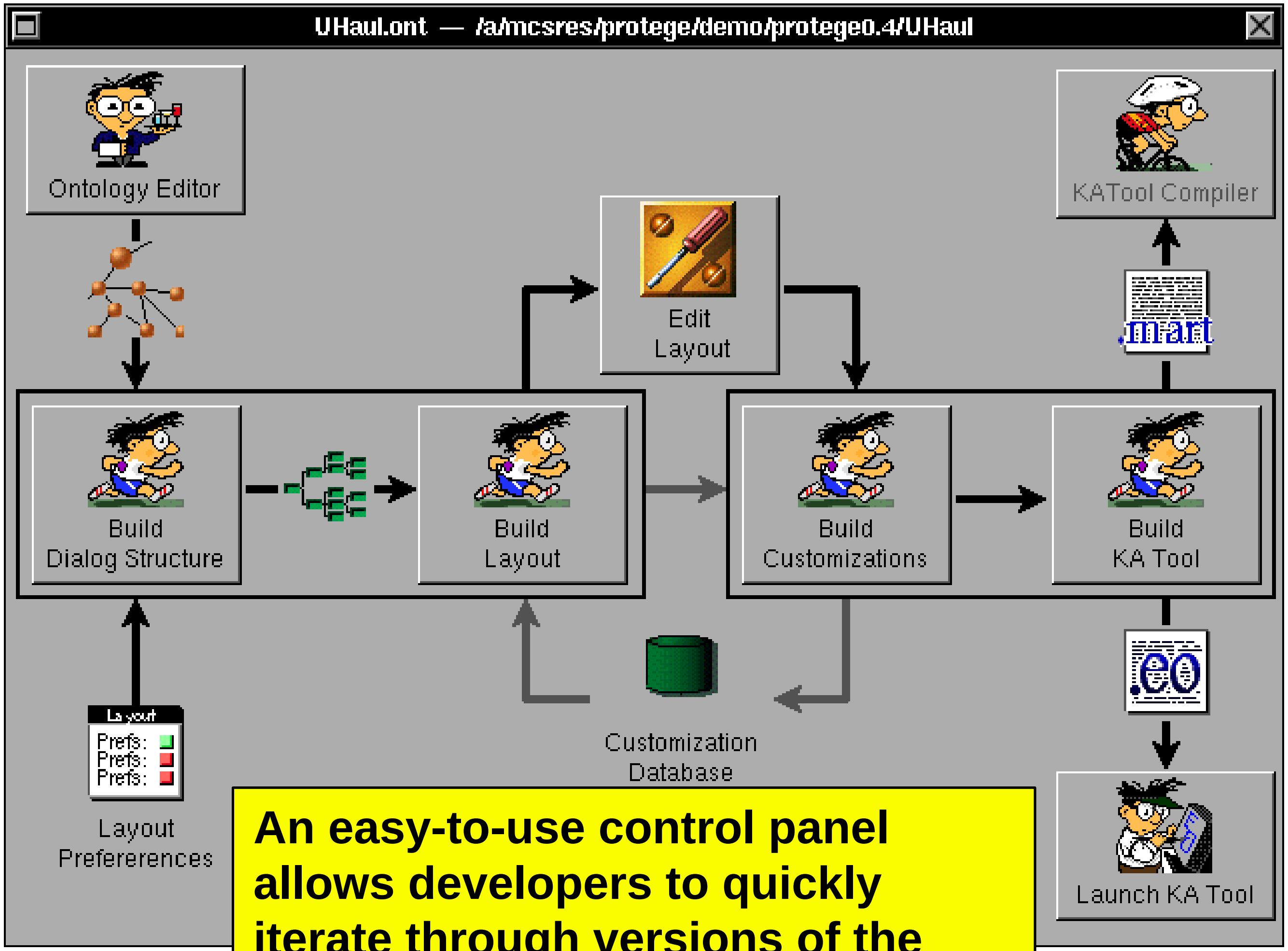

```
(slot employees (type instance)
  (allowed-classes person))
```



DASH Architecture

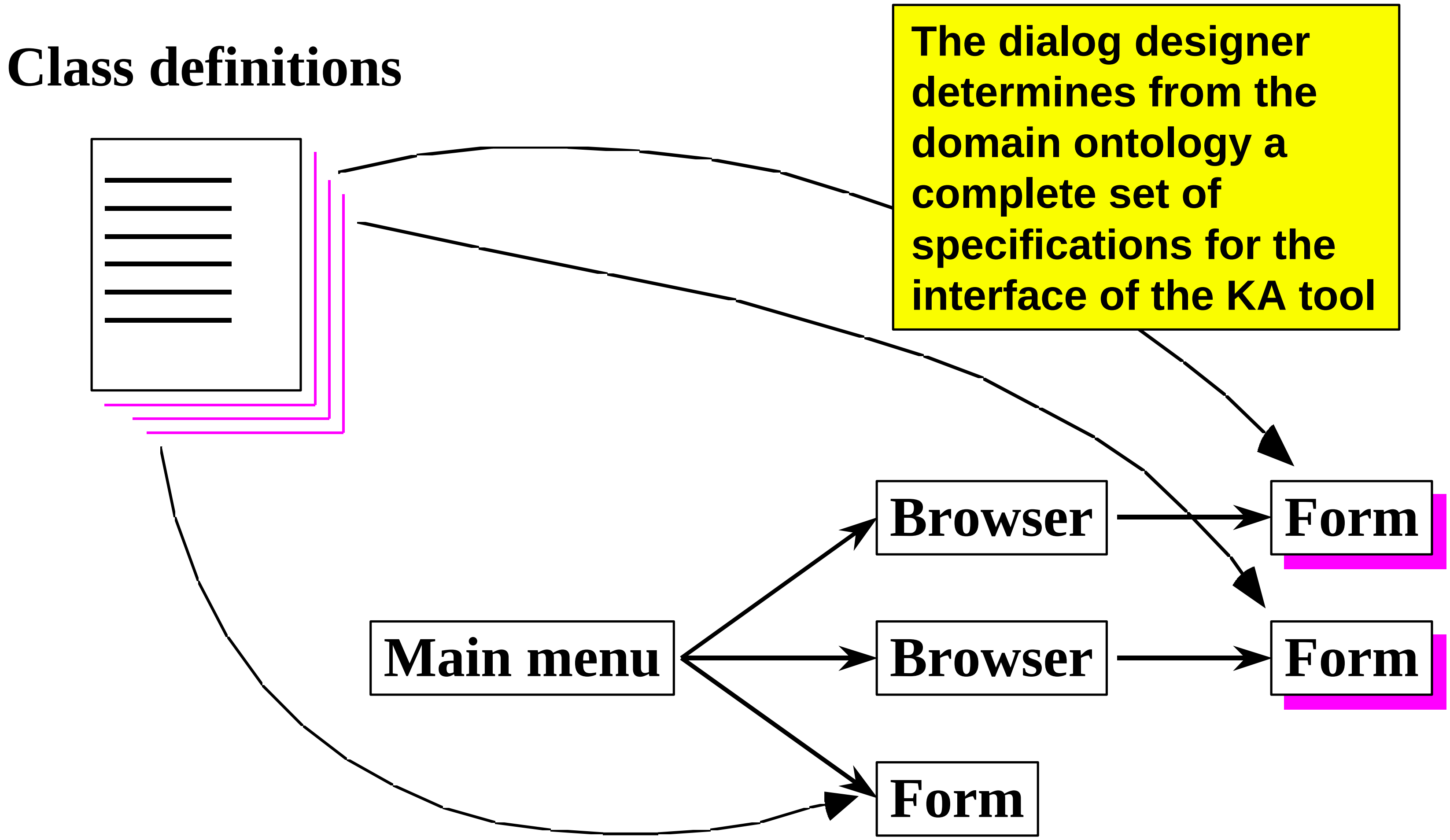


The DASHboard

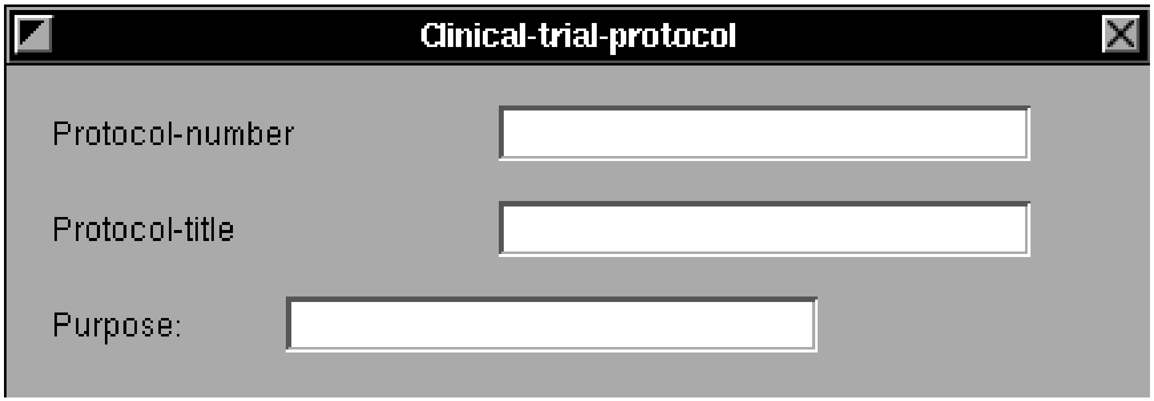


An easy-to-use control panel allows developers to quickly iterate through versions of the generated KA tool

DASH: Dialog Designer



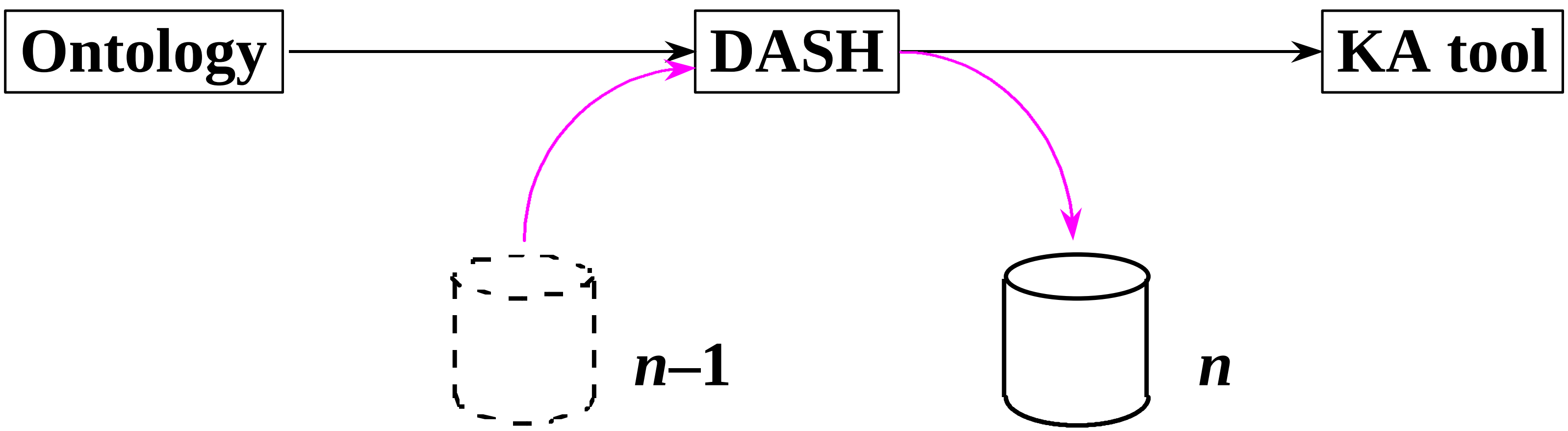
DASH: Custom Adjustments



An interface builder supports changes by direct manipulation to the layout of the generated KA tool

DASH: Persistent Custom Adjustments

What will happen to my knowledge-acquisition tool if I make changes to the ontology?



Customization databases

Use of DASH

Examples of problem domains

- » Sisyphus room-assignment
- » Airport gate allocation
- » Hospital bed assignment
- » Rental equipment (UHaul)
- » Elevator configuration (VT/SALT)
- » Clinical trial protocol
- » Ribosome topology
- » Internist/QMR
- » Meta Land (ontology of ontology)

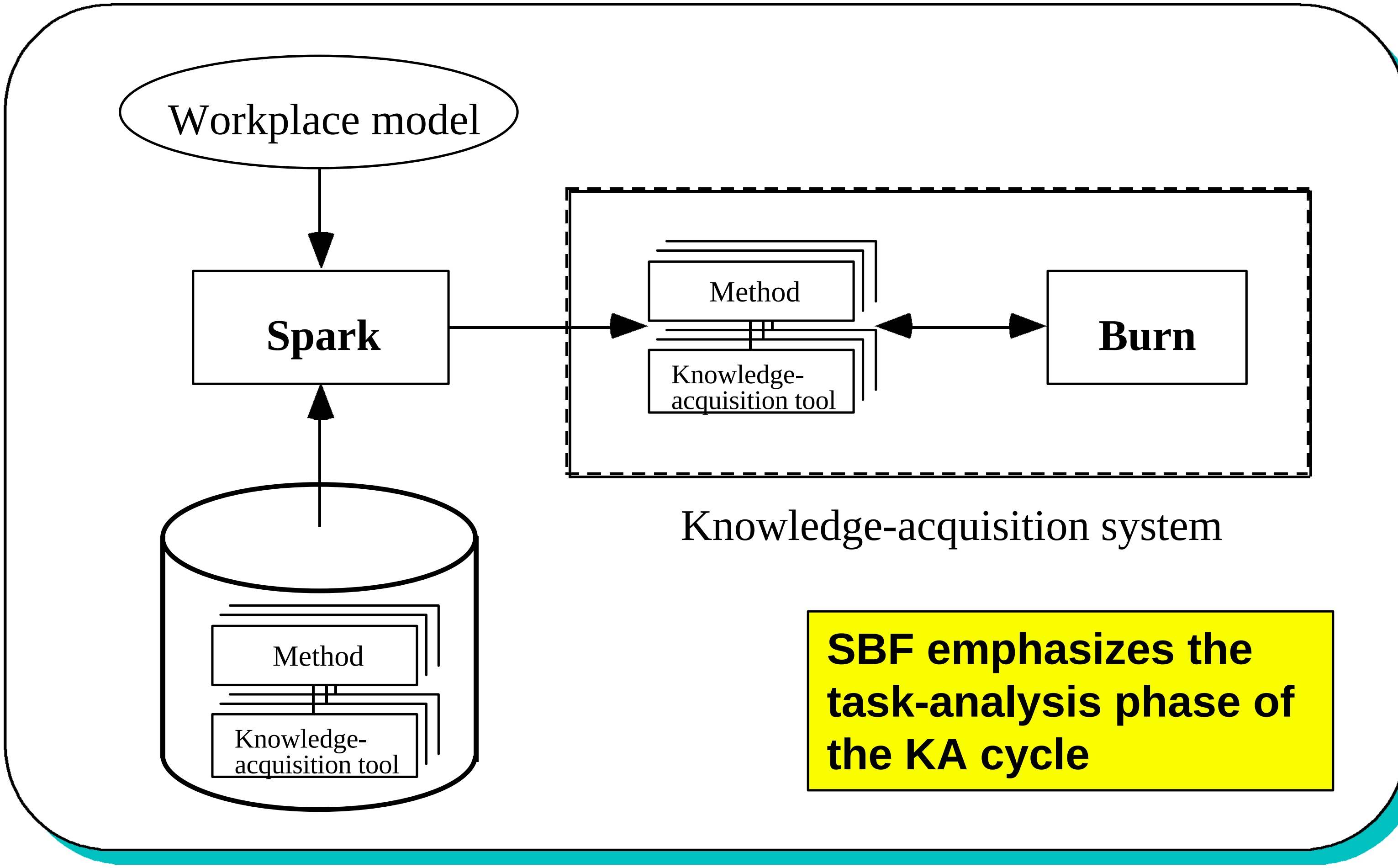
PROTÉGÉ-II Summary

- **Benefits:**
  - » Reusable problem-solving methods
  - » Reusable domain ontologies
  - » Ontology-based generation of knowledge-acquisition tools
  - » Support for early prototyping
- **Shortcomings:**
  - » No automated support for task analysis
- **Users: developers of knowledge-based systems**

Spark, Burn, and FireFighter (SBF)

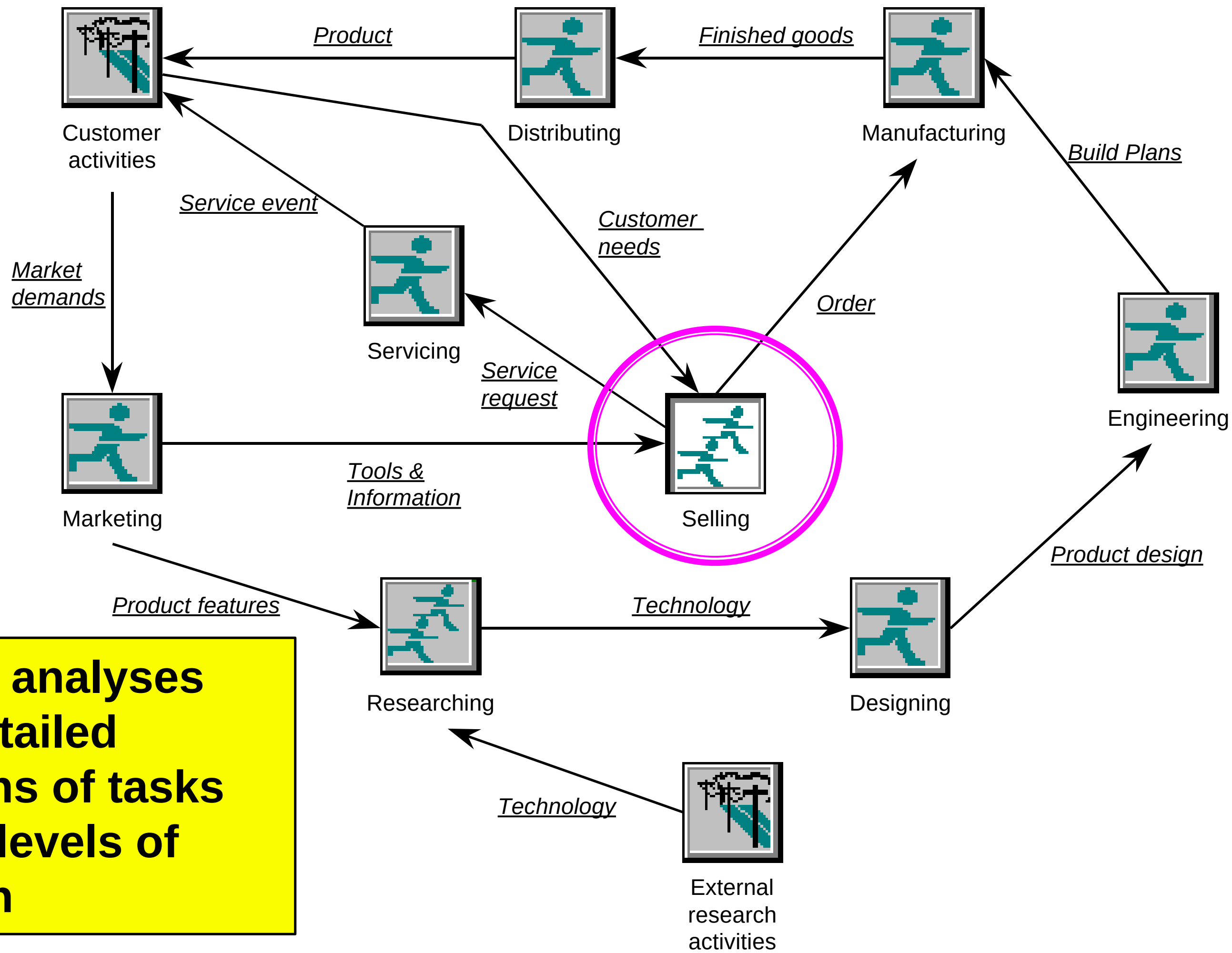
- **Spark**
  - » **Uses a workplace model as the basis for selection of a problem-solving methods from a library**
  - » **Configures knowledge-acquisition tools associated with the methods selected**
- **Burn**
  - » **Runtime system for knowledge-acquisition tools**
  - » **Controls the knowledge-acquisition session**
- **FireFighter**
  - » **Debugging tool**

SBF Architecture



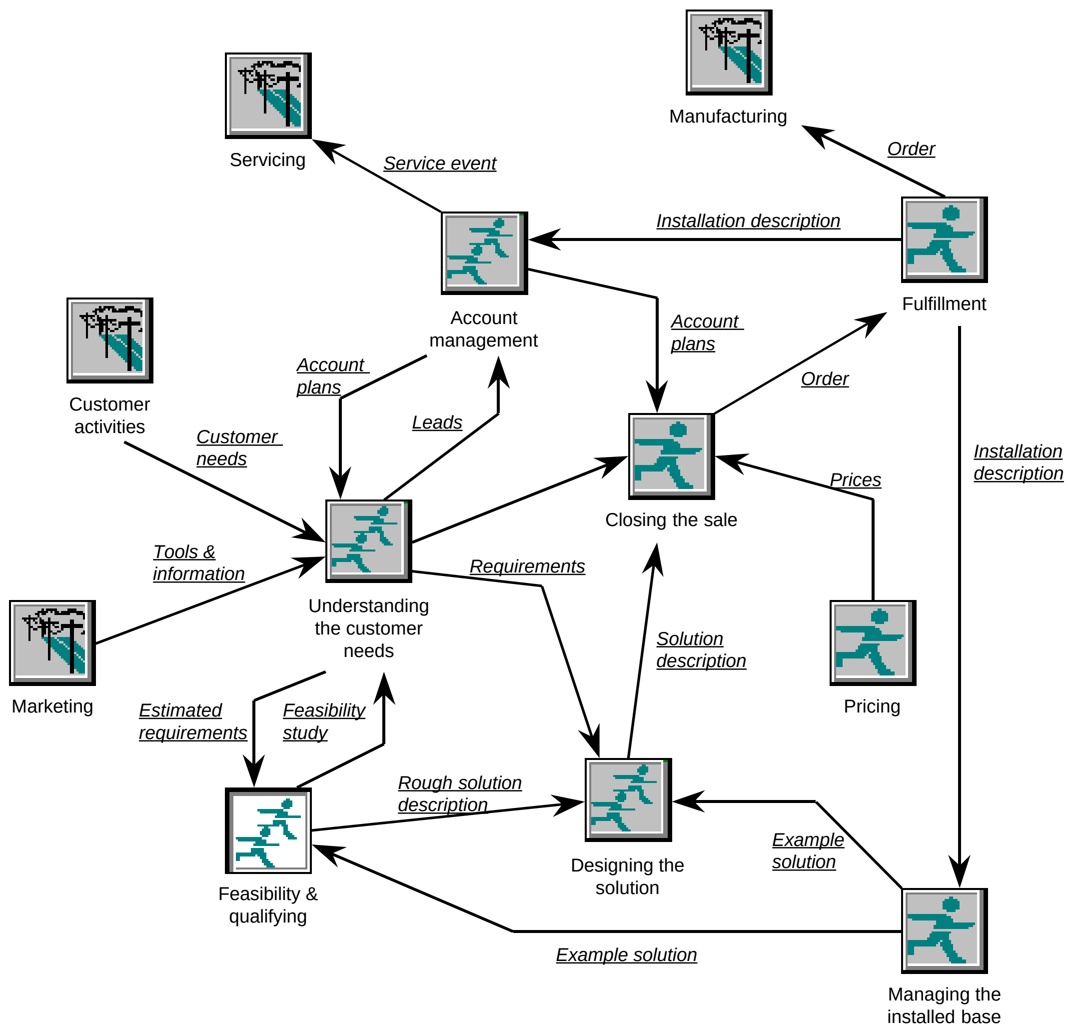
SBF emphasizes the task-analysis phase of the KA cycle

SBF: Workplace Analysis



Workplace analyses provide detailed descriptions of tasks at various levels of abstraction

SBF: Workplace Analysis (2)



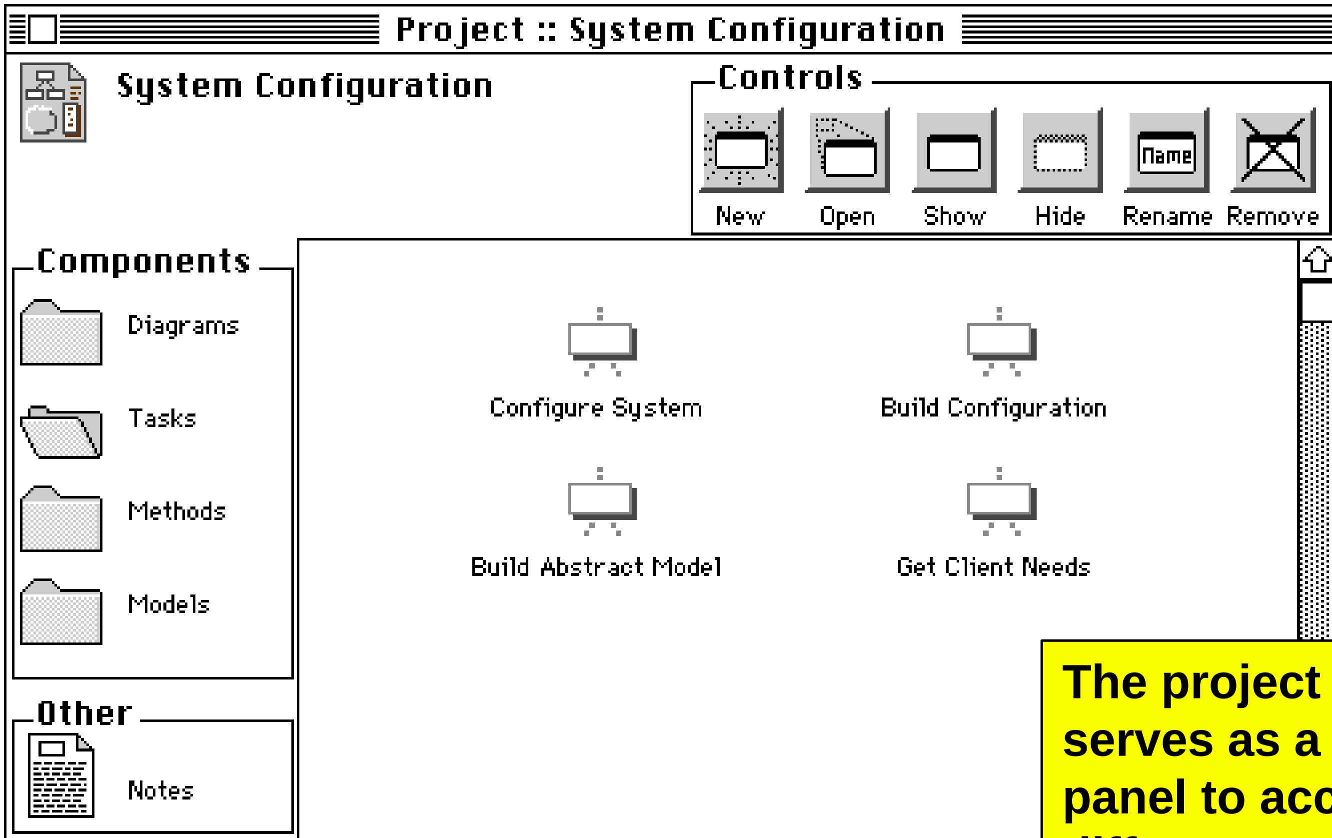
SBF Summary

- **Benefits:**
  - » **Comprehensive support for task analysis**
  - » **Reusable problem-solving methods**
- **Shortcomings:**
  - » **One KA tool per method (method-specific tools)**
- **Users: non-programmers**

KREST

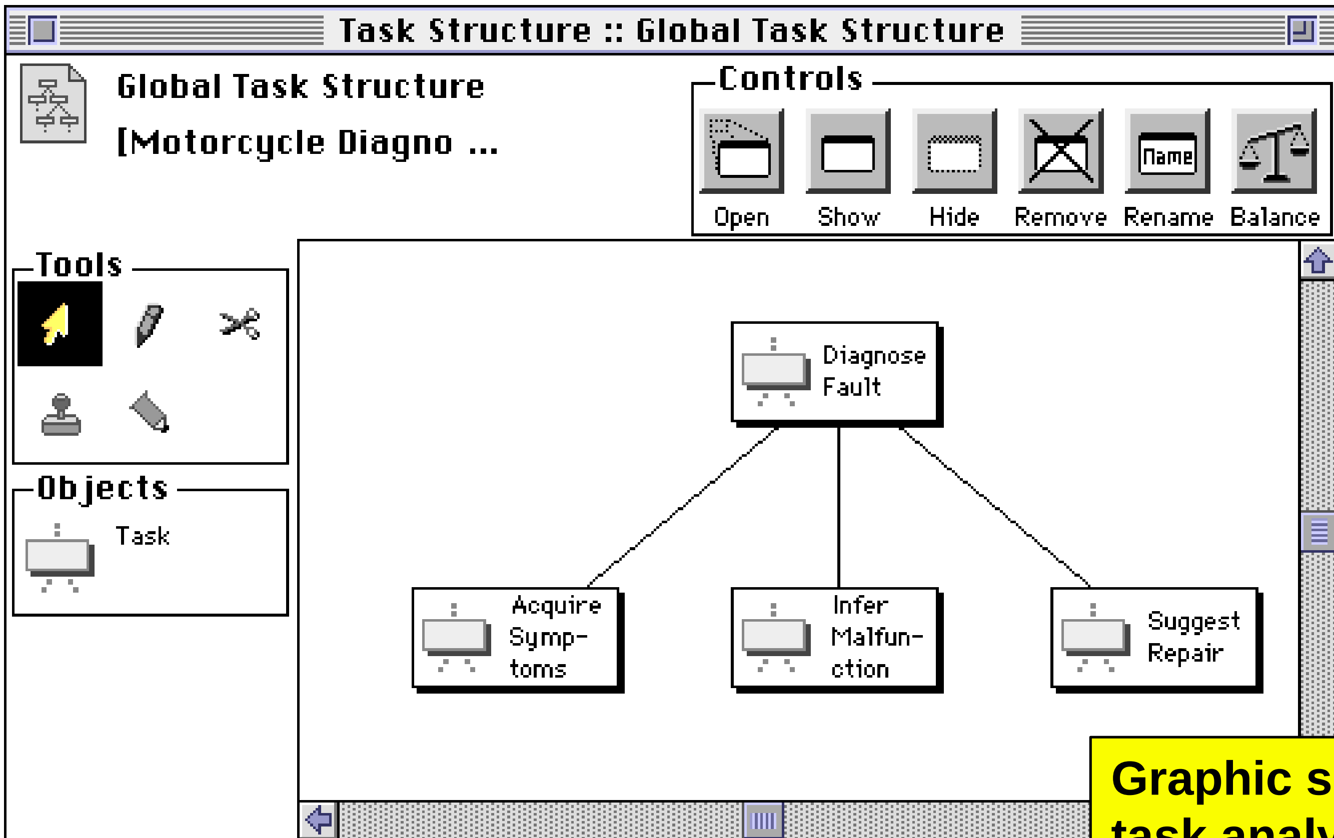
- A KE environment for configuration of applications through sharing and reuse of components
- Targeted to non-programmers
- Based on “componential” methodology
  - » Systems are made up of reusable knowledge components
- Establishes framework for knowledge-level modeling
- Explicit linking between knowledge level and symbol-level components

KREST: Project Window



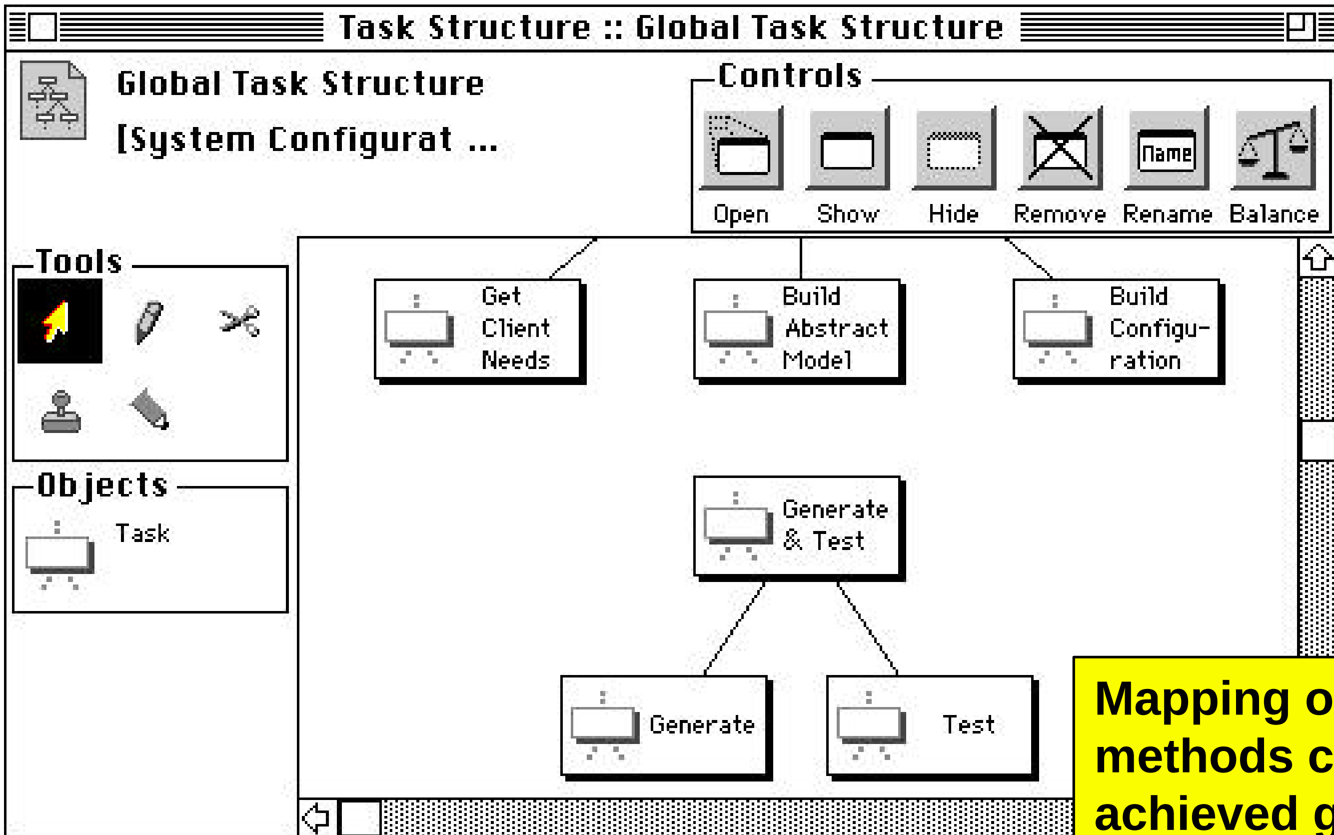
The project window serves as a control panel to access the different parts of the project

KREST: Task Structures



Graphic support for task analysis

KREST: Configuring Applications



Mapping of tasks to methods can be achieved graphically

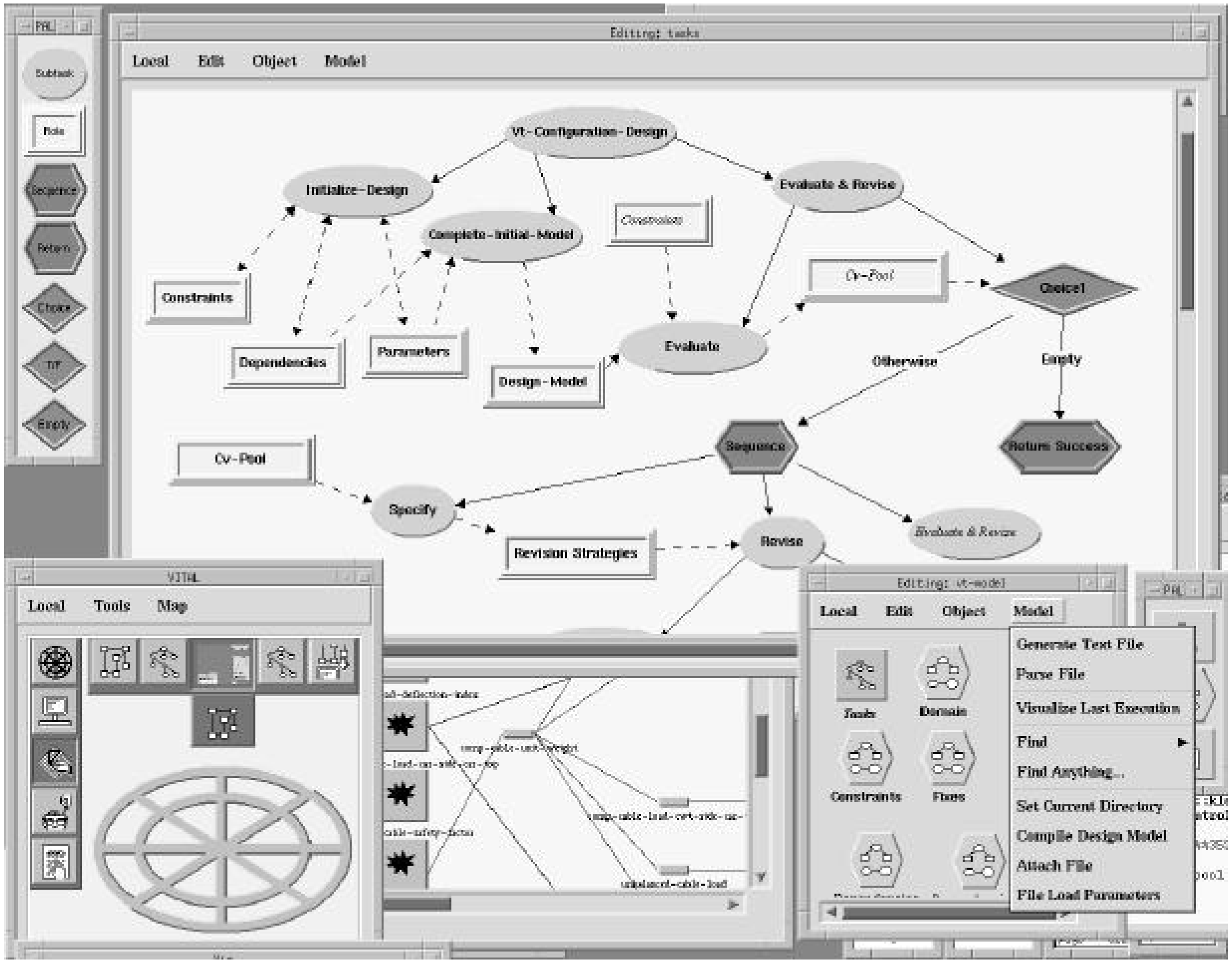
KREST Summary

- **Benefits:**
  - » **Integrated graphical environment**
  - » **Established user community**
- **Shortcomings:**
  - » **Work required at the symbol level**
- **Users: non-programmers**

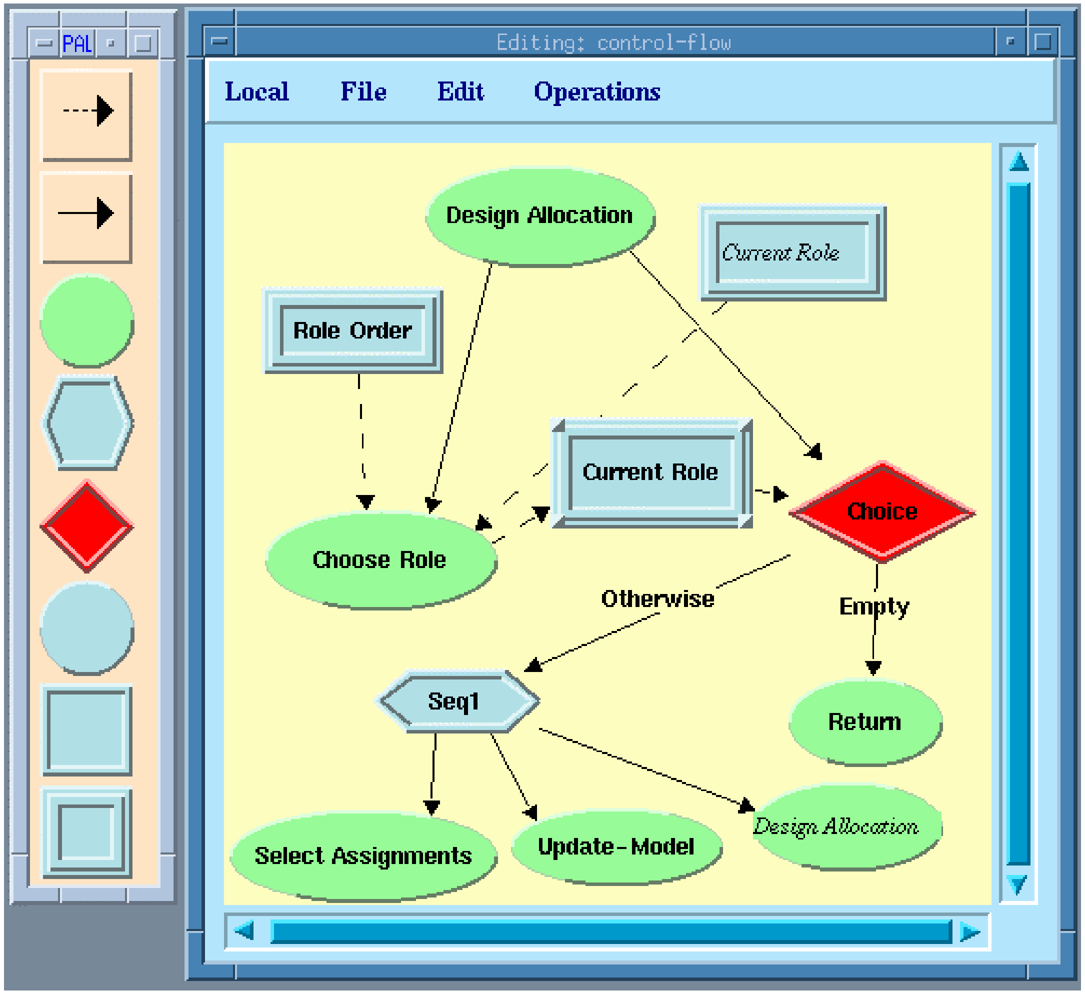
VITAL

- **Knowledge-engineering workbench**
  - » Methodological and tool support for structured KBS development
  - » Support for project management
  - » Model refinement at several levels of abstractions (cf. KADS).
  - » Integration of multiple KBS and SE technologies (e.g., KA, ML, Groupware, Software, and Visualization)
- **Main techniques::**
  - » Generalized Directive Model (GDM) Analysis (Coarse-grained models) (van Heijst et al., 1992; Motta et al., in press)
  - » Task Structure Analysis (Fine-grained models)
  - » The resulting, fine-grained, model is then linked to symbol-level structures

The VITAL Workbench



Control Flow in VITAL



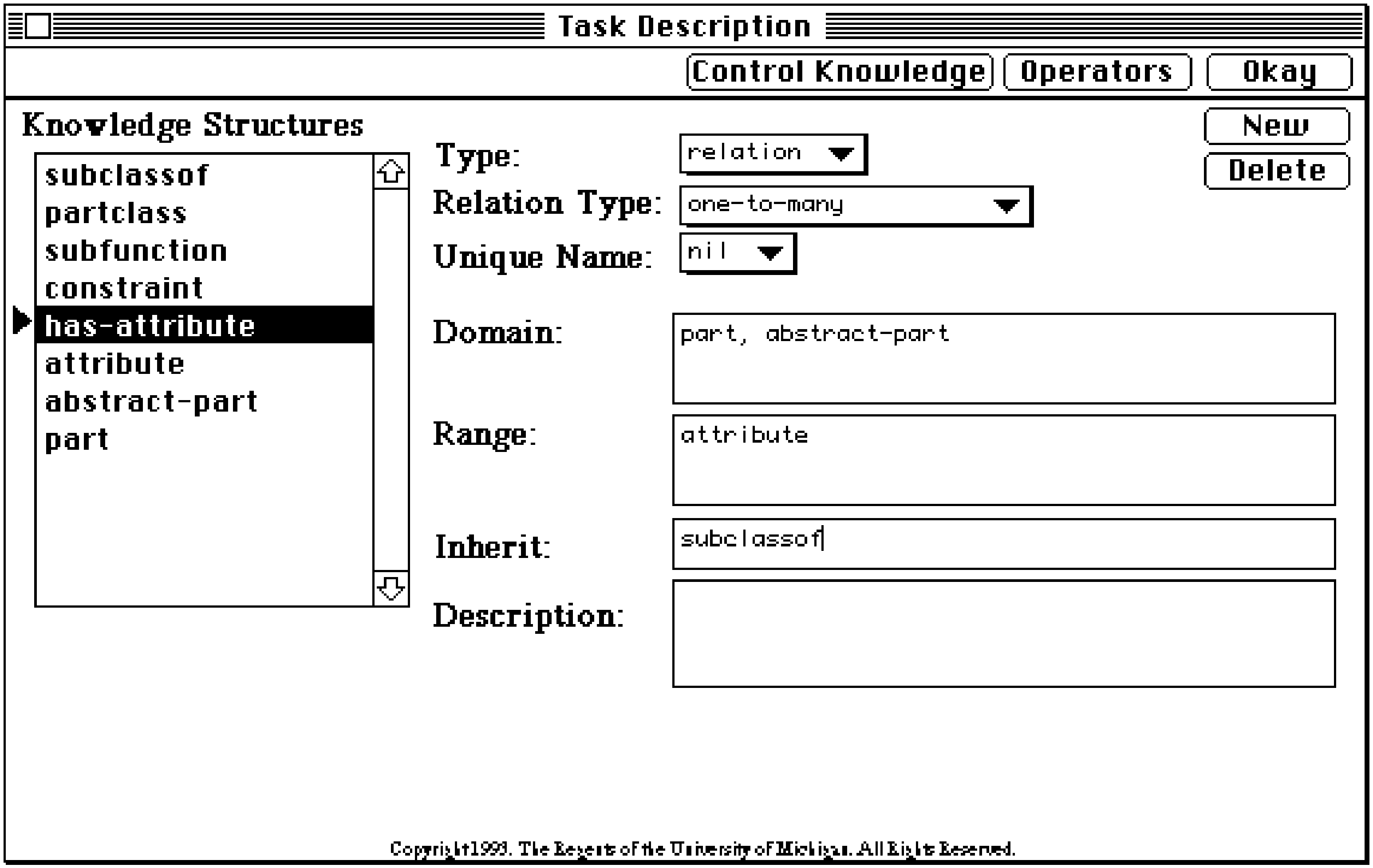
VITAL Summary

- **Benefits:**
  - » **Extensive program visualization capabilities**
  - » **Use of defined methodologies**
  - » **Support of software engineering principles**
- **Shortcomings:**
  - » **Scope may be too broad**
- **Users: developers of knowledge-based systems**

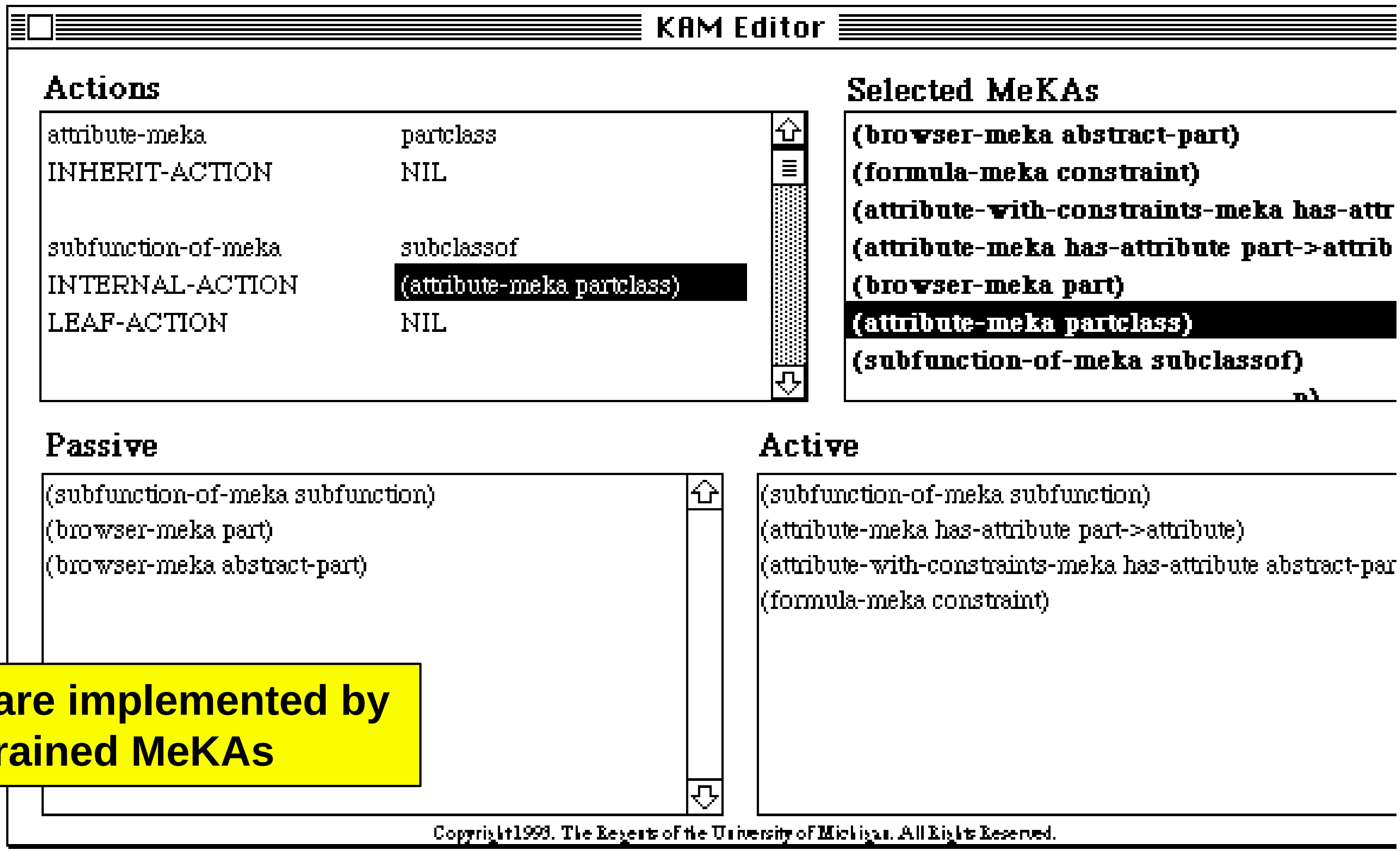
DIDS

- **Domain-Independent Design System**
- **DIDS supports**
  - » **Knowledge-level task description**
  - » **Process model**
  - » **Knowledge-acquisition model**
- **Design and configuration tasks: Support for method reuse and method-oriented knowledge-acquisition**
- **Knowledge-acquisition components: *mechanisms for knowledge acquisition* (MeKAs) and *knowledge-acquisition methods* (KAMs)**
  - » **Library of reusable MeKAs**
  - » **Individually, most MeKAs are symbol-level tools**

Task Description in DIDS: Specification of Relations

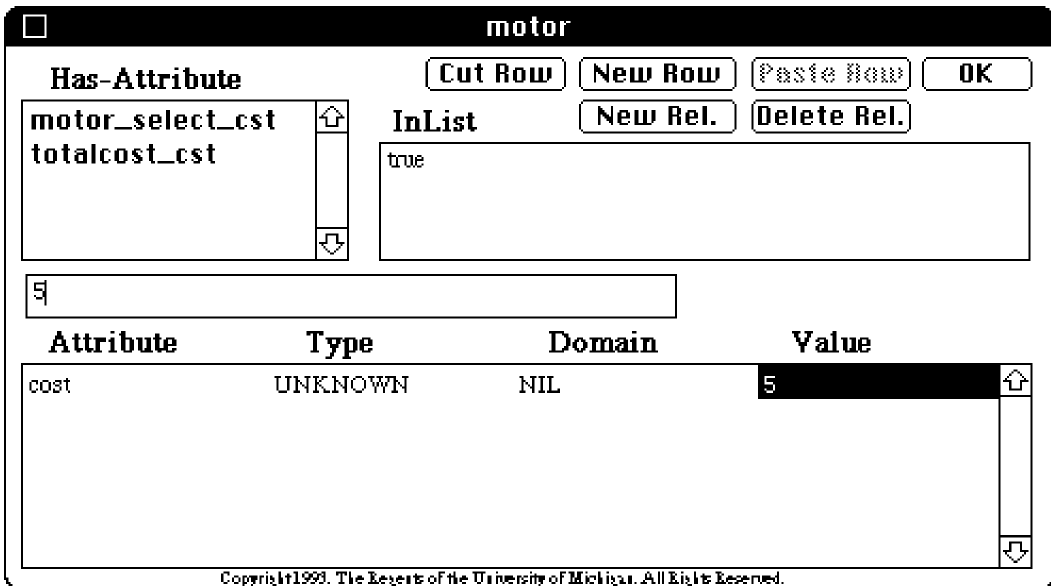


DIDS: Knowledge-Acquisition Method (KAM)

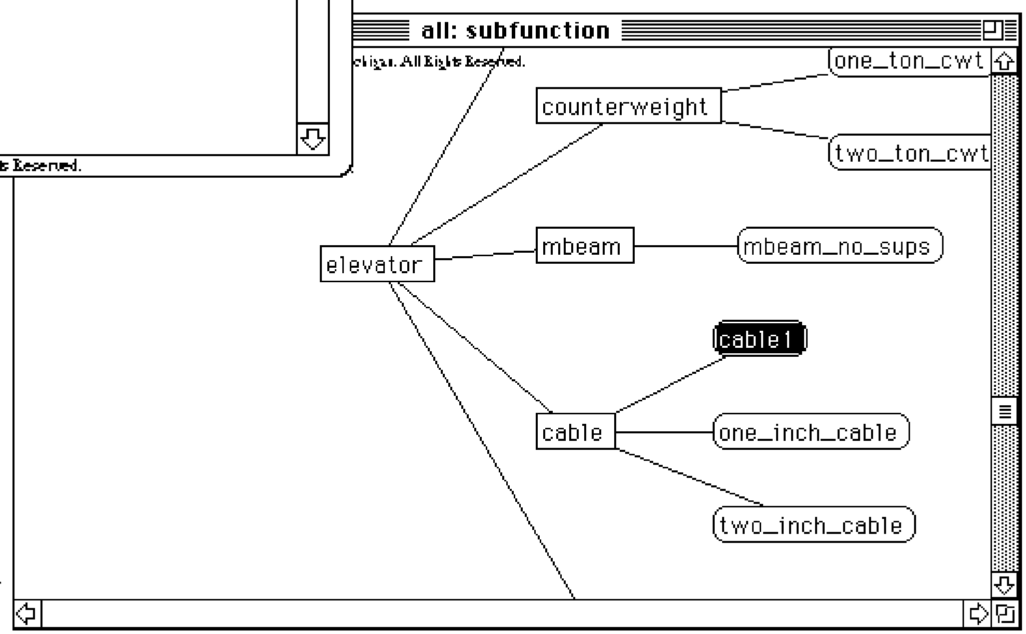


KAMs are implemented by finer-grained MeKAs

DIDS: MeKAs



Attribute specification



Graphical Node–Link Diagram

DIDS Summary

- **Benefits:**
  - » **Extensive support for design and configuration tasks**
  - » **Library of reusable MeKAs**
- **Shortcomings:**
  - » **Support limited to design and configuration tasks**
- **Users: developers of knowledge-based systems**

Summary: KE Environments

Ontology-driven knowledge engineering

» **PROTÉGÉ-II: DASH and MART**

» **Method ontologies**

Method-driven knowledge engineering

» **SBF: Spark and Burn**

» **VITAL: Generalized directive model (GDM)**

» **DIDS: Knowledge-acquisition method (KAM) and mechanism for knowledge acquisition (MeKA)**

Agenda

- Knowledge engineering concepts
- Current trends in knowledge-based development
- Break
- Case Studies
- Incorporating knowledge engineering tools into software projects
- Summary: Lessons learned and future directions
- Questions

Knowledge Engineering Environments and the Software Engineering Cycle

- **Knowledge environments:**
  - » **Do not cover complete software cycles but…**
  - » **Cover very well non-traditional phases (e.g., domain modeling)**
- **Non-automated, manual programming labor must be expected**
- **The key to success is working out a good fit for a KE environment in your existing life cycle**

A Cooperating Scenario

- **Build an application via intermediate files:**
  - » **Construct domain model in Protégé-II**
  - » **Generate knowledge-acquisition tool**
  - » **Populate knowledge base**
  - » **Use a translator to store knowledge base in a database. Build data model from domain model**
  - » **Develop application to access and manipulate data base**

KE Environments Applicability Guidelines

- **Identify the capabilities of the KE Environment**
  - » **Use the material from this tutorial**
  - » **Ask provider for demo**
- **Redesign your software life cycle**
  - » **Match KE environment functionality to current tasks (esp. manual ones)**
  - » **Determine processes for phase transitions (e.g., intermediate files)**
- **Analyze costs and benefits**
  - » **Study feasibility for a single application**
  - » **Examine potential reusability benefits**

Feasibility Considerations

- **Target application or KBS**
  - » New applications require much initial modeling
  - » Existing applications constrain selection of KE Environments
  - » Automated support decreases for complex knowledge bases
- **Knowledge representation**
  - » KE environment may introduce knowledge representation incompatibilities
  - » Are translators available?
- **Reusable components**
  - » Can the KE Environment be used in other projects?
  - » Are any by-products reusable?
  - » What reusable components does the KE environment provide? (e.g., problem-solving methods, domain models)

Summary: Incorporating KE Tools Into Software Projects

Success depends on a good life cycle match

Don’t overlook reusability benefits

Selection and design guidelines for KE tools are similar!

Agenda

- Knowledge engineering concepts
- Current trends in knowledge-based development
- Break
- Case Studies
- Incorporating knowledge engineering tools into software projects
- Summary: Lessons learned and future directions
- Questions

Advantages of Reusable-Component Knowledge Engineering

- Integrated environments for KA and KBS development
- Life cycle support
- Lower development costs
- High levels of automation
- Well-defined methodologies for KBS development

Challenges of Current Technology

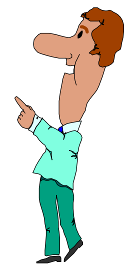
- **Limited availability of problem-solving methods**
- **Indexing of libraries not addresses**
- **Existing environments are generally strong in only some phases of the life cycle**
- **Optimal granularity of methods and ease of reusability have not been established**
- **Migration to Internet**

Trends and Short-Term Future Developments

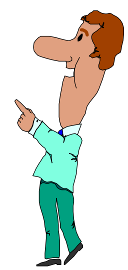
- **Distributed environments**
- **Increased availability of problem-solving methods**
- **Increased availability of domain ontologies**
- **Reliable indexing systems for method selection**
- **Tools for library maintenance**
- **Increased emphasis and automated support for reusability**

Take-home messages

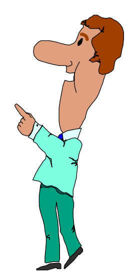
Modern KBS development requires concurrent development of the KBS and its KA tool



KE environments are the answer to the need of concurrent, comprehensive development



Reuse, reuse, reuse....and save



Agenda

- Knowledge engineering concepts
- Current trends in knowledge-based development
- Break
- Case Studies
- Incorporating knowledge engineering tools into software projects
- Summary: Lessons learned and future directions
- Questions

Questions

?

