<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Propositional Logic</td>
</tr>
<tr>
<td>3</td>
<td>Predicate Logic</td>
</tr>
<tr>
<td>4</td>
<td>Reasoning</td>
</tr>
<tr>
<td>5</td>
<td>Search Methods</td>
</tr>
<tr>
<td>6</td>
<td><strong>CommonKADS</strong></td>
</tr>
<tr>
<td>7</td>
<td>Problem-Solving Methods</td>
</tr>
<tr>
<td>8</td>
<td>Planning</td>
</tr>
<tr>
<td>9</td>
<td>Software Agents</td>
</tr>
<tr>
<td>10</td>
<td>Rule Learning</td>
</tr>
<tr>
<td>11</td>
<td>Inductive Logic Programming</td>
</tr>
<tr>
<td>12</td>
<td>Formal Concept Analysis</td>
</tr>
<tr>
<td>13</td>
<td>Neural Networks</td>
</tr>
<tr>
<td>14</td>
<td>Semantic Web and Services</td>
</tr>
</tbody>
</table>
Agenda

1. Motivation
2. Technical solution, illustrations and extensions
   1. Overview of CommonKADS
   2. Knowledge model components
   3. Template knowledge models
   4. Knowledge model construction
   5. Knowledge elicitation techniques
3. Example
4. Summary
5. References


And slides are partly based on the CommonKads Course http://www.commonkads.uva.nl
MOTIVATION
Knowledge engineering

• Process of
  – eliciting,
  – structuring,
  – formalizing,
  – operationalizing
• information and knowledge involved in a knowledge-intensive problem domain,
• in order to construct a program that can perform a difficult task adequately
Knowledge engineering problems

- Complex information and knowledge is difficult to observe
- Experts and other sources differ
- Multiple representations:
  - textbooks
  - graphical representations
  - skills
• Knowledge is valuable and often outlives a particular implementation
  – knowledge management
• Errors in a knowledge-base can cause serious problems
• Heavy demands on extendibility and maintenance
  – changes over time
TECHNICAL SOLUTION AND ILLUSTRATIONS
Overview of CommonKADS
CommonKADS principles

• CommonKADS: a comprehensive methodology for KBS development
• Knowledge engineering is *not* some kind of `mining from the expert's head', but consists of constructing different aspect models of human knowledge
• The knowledge-level principle: in knowledge modeling, first concentrate on the conceptual structure of knowledge, and leave the programming details for later
• Knowledge has a stable internal structure that is analyzable by distinguishing specific knowledge types and roles.
CommonKADS Terminology

- **Domain**
  - some area of interest
    - banking, food industry, photocopiers, car manufacturing

- **Task**
  - something that needs to be done by an agent
    - monitor a process; create a plan; analyze deviant behavior

- **Agent**
  - the executor of a task in a domain
    - typically either a human or some software system

- **Application**
  - The context provided by the combination of a task and a domain in which this task is carried out by agents

- **Application domain**
  - The particular area of interest involved in an application

- **Application task**
  - The (top-level) task that needs to be performed in a certain application
CommonKADS Terminology

• knowledge system (KS)
  – system that solves a real-life problem using knowledge about the application domain and the application task

• expert system
  – knowledge system that solves a problem which requires a considerable amount of expertise, when solved by humans.
CommonKADS Model Set

Context
- Organization Model
- Task Model
- Agent Model

Concept
- Knowledge Model
- Communication Model

Artefact
- Design Model
Model Set Overview (1)

- **Organization model**
  - supports analysis of an organization,
  - Goal: discover problems, opportunities and possible impacts of KBS (knowledge-based system) development.

- **Task model**
  - describes tasks that are performed or will be performed in the organizational environment

- **Agent model**
  - describes capabilities, norms, preferences and permissions of agents (agent = executor of task).
• Knowledge model
  – gives an implementation-independent description of knowledge involved in a task.

• Communication model
  – models the communicative transactions between agents.

• Design model
  – describes the structure of the system that needs to be constructed.
Models exist in various forms

- **Model template**
  - predefined, fixed structure, can be configured

- **Model instance**
  - objects manipulated during a project.

- **Model versions**
  - versions of a model instance can exist.

- **Multiple model instances**
  - separate instances can be developed
  - example: "current" and "future" organization
The Product

- **Instantiated models**
  - represent the important aspects of the environment and the delivered knowledge based system.

- **Additional documentation**
  - information not represented in the filled model templates (e.g. project management information)

- **Software**
Roles in knowledge-system development

- knowledge provider
- knowledge engineer/analyst
- knowledge system developer
- knowledge user
- project manager
- knowledge manager

many-to-many relations between roles and people
Knowledge provider/specialist

- “traditional” expert
- person with extensive experience in an application domain
- can provide also plan for domain familiarization
  - “where would you advise a beginner to start?”
- inter-provider differences are common
- need to assure cooperation
Knowledge engineer

- specific kind of system analyst
- should avoid becoming an "expert"
- plays a liaison function between application domain and system
Knowledge-system developer

- person that implements a knowledge system on a particular target platform
- needs to have general design/implementation expertise
- needs to understand knowledge analysis
  - but only on the “use”-level
Knowledge user

• interact with the prospective system or are affected indirectly by the system
• Level of skill/knowledge is important factor
• May need extensive interacting facilities
  – explanation
• His/her work is often affected by the system
  – consider attitude / active role
Project manager

- responsible for planning, scheduling and monitoring development work
- liaises with client
- typically medium-size projects (4-6 people)
- profits from structured approach
Knowledge manager

• background role
• monitors organizational purpose of
  – system(s) developed in a project
  – knowledge assets developed/refined
• initiates (follow-up) projects
• should play key role in reuse
• may help in setting up the right project team
Roles in knowledge-system development

- Knowledge manager:
  - Defines knowledge strategy
  - Initiates knowledge development projects
  - Facilitates knowledge distribution

- Knowledge provider/specialist:
  - Elicits knowledge from
  - Validates

- Knowledge user:
  - Uses

- Knowledge engineer/analyst:
  - Elicits requirements from
  - Manages

- Project manager:
  - Manages

- Knowledge system developer:
  - Designs & implements
  - Delivers analysis models to

KS (Knowledge System)
Knowledge model components
Knowledge model

- specialized tool for specification of knowledge-intensive tasks
- abstracts from communication aspects
- real-world oriented
- reuse is central theme
Knowledge categories

• Domain knowledge
  – relevant domain knowledge and information
  – static

• Inference knowledge
  – basic reasoning steps that can be made in the domain knowledge and are applied by tasks

• Task knowledge
  – goal-oriented
  – functional decomposition
Knowledge Categories: domain knowledge

- **domain schema**
  - schematic description of knowledge and information types
  - defined through domain constructs
- **knowledge base**
  - set of knowledge instances
Constructs for domain schema

• Concept
  – cf. object class (without operations)
• Relation
  – cf. association
• Attribute
  – primitive value
• Rule type
  – introduces expressions
Example: car concepts

CONCEPT gas dial;
  ATTRIBUTES:
    value: dial-value;
END CONCEPT gas-dial;

VALUE-TYPE dial-value;
  VALUE-LIST: {zero, low, normal};
  TYPE: ORDINAL;
END VALUE-TYPE dial-value;

CONCEPT fuel-tank;
  ATTRIBUTES:
    status: {full, almost-empty, empty};
END CONCEPT fuel-tank;
Modelling rules

- knowledge analysis is focused on finding rules with a common structure
- a rule as an instance of a rule type
- models a relation between expressions about feature values (e.g. attribute values)
  \[
  \text{gas-dial.value = zero} \implies \text{fuel-tank.status = empty}
  \]
- models set of real-world “rules” with a similar structure
Example rule type

- **person**
  - name: string
  - income: integer

- **loan**
  - amount: integer
  - Interest-rate: number

**Loan constraint**

- $\text{person.income} \leq 10,000$
  - RESTRICTS
  - $\text{loan.amount} \leq 2,000$

- $\text{person.income} > 10,000$ AND $\text{person.income} \leq 20,000$
  - RESTRICTS
  - $\text{loan.amount} \leq 3,000$
Rule type structure

• `<antecedent> <connection-symbol> <consequent>`
• example rule:
  
  fuel-supply.status = blocked
  CAUSES
  gas-in-engine.status = false;
• flexible use for almost any type of dependency
  – multiple types for antecedent and consequent
Inference knowledge

- describes the lowest level of functional decomposition
- basic information-processing units:
  - inference => reasoning
  - transfer function => communication with other agents
- why special status?
  - indirectly related to domain knowledge
  - enables reuse of inference
Example inference: cover

**dynamic input role**

- **complaint**: my car does not start

**inference**

- **cover**

**dynamic output role**

- **hypothesis**: fuel tank is empty

**causal model**

- **static role**

  - fuel tank is empty leads to lack of gas in engine
  - if there is no gas in the engine, then the car does not start
Task knowledge

- describes goals
  - assess a mortgage application in order to minimize the risk of losing money
  - find the cause of a malfunction of a photocopier in order to restore service.
  - design an elevator for a new building.

- describes strategies that can be employed for realizing goals.

- typically described in a hierarchical fashion
Task

- Description of the input/output
- Main distinction with traditional functions is that the data manipulated by the task are (also) described in a domain-independent way.
  - example, the output of a medical diagnosis task would not be a “disease” but an abstract name such as “fault category”
Template knowledge models
Lessons

- Knowledge models partially reused in new applications
- Type of task = main guide for reuse
- Catalog of task templates
The need for reuse

- prevent "re-inventing the wheel"
- cost/time efficient
- decreases complexity
- quality-assurance
Task template

- reusable combination of model elements
  - (provisional) inference structure
  - typical control structure
  - typical domain schema from task point-of-view
- specific for a task type
- supports top-down knowledge modeling
Analytic versus synthetic tasks

• analytic tasks
  – system pre-exists
    • it is typically not completely "known"
  – input: some data about the system,
  – output: some characterization of the system

• synthetic tasks
  – system does not yet exist
  – input: requirements about system to be constructed
  – output: constructed system description
Task hierarchy

- Knowledge intensive task
  - Analytic task
    - Classification
    - Diagnosis
    - Prediction
    - Assessment
    - Monitoring
  - Synthetic task
    - Design
    - Planning
    - Assignment
    - Modeling
    - Scheduling
    - Configuration design
Structure of template description in catalog

- General characterization
  - typical features of a task
- Default method
  - roles, sub-functions, control structure, inference structure
- Typical variations
  - frequently occurring refinements/changes
- Typical domain-knowledge schema
  - assumptions about underlying domain-knowledge structure
Classification

• establish correct class for an object
• object should be available for inspection
  – "natural" objects
• examples: rock classification, apple classification
• terminology: object, class, attribute, feature
• one of the simplest analytic tasks; many methods
• other analytic tasks: sometimes reduced to classification problem especially diagnosis
Assessment

• find decision category for a case based on domain-specific norms.
• typical domains: financial applications (loan application), community service
• terminology: case, decision, norms
• some similarities with monitoring
  – differences:
    • timing: assessment is more static
    • different output: decision versus discrepancy
Diagnosis

• find fault that causes system to malfunction
  – example: diagnosis of a copier
• terminology:
  – complaint/symptom, hypothesis, differential, finding(s)/evidence, fault
• nature of fault varies
  – state, chain, component
• should have some model of system behavior
  – default method: simple causal model
• sometimes reduced to classification task
  – direct associations between symptoms and faults
• automation feasible in technical domains
Monitoring

• analyze ongoing process to find out whether it behaves according to expectations
• terminology:
  – parameter, norm, discrepancy, historical data
• main features:
  – dynamic nature of the system
  – cyclic task execution
• output "just" discrepancy => no explanation
• often: coupling monitoring and diagnosis
  – output monitoring is input diagnosis
Prediction

- analytic task with some synthetic features
- analyses current system behavior to construct description of a system state at future point in time.
- example: weather forecasting
- often sub-task in diagnosis
- also found in knowledge-intensive modules of teaching systems e.g. for physics.
- inverse: retrodiction: big-bang theory
Synthesis

• Given a set of requirements, construct a system description that fulfills these requirements

<table>
<thead>
<tr>
<th>requirements</th>
<th>constraints &amp; preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>(external)</td>
<td>(internal)</td>
</tr>
<tr>
<td>soft requirement</td>
<td>preference</td>
</tr>
<tr>
<td>&quot;fast system&quot;</td>
<td>&quot;prefer cheapest component&quot;</td>
</tr>
<tr>
<td>hard requirement</td>
<td>constraint</td>
</tr>
<tr>
<td>&quot;price lower than $2,000&quot;</td>
<td>&quot;P 166 processor requires 16Mb&quot;</td>
</tr>
</tbody>
</table>
“Ideal” synthesis method

- Operationalize requirements
  - preferences and constraints
- Generate all possible system structures
- Select sub-set of valid system structures
  - obey constraints
- Order valid system structures
  - based on preferences
Design

• synthetic task
• system to be constructed is physical artifact
• example: design of a car
• can include creative design of components
• creative design is too hard a nut to crack for current knowledge technology
• sub-type of design which excludes creative design => configuration design
Configuration design

• given predefined components, find assembly that satisfies requirements + obeys constraints
• example: configuration of an elevator; or PC
• terminology: component, parameter, constraint, preference, requirement (hard & soft)
• form of design that is well suited for automation
• computationally demanding
Assignment

- create mapping between two sets of objects
  - allocation of offices to employees
  - allocation of airplanes to gates
- mapping has to satisfy requirements and be consistent with constraints
- terminology
  - subject, resource, allocation
- can be seen as a “degenerate” form of configuration design
• shares many features with design
• main difference: "system" consists of activities plus time dependencies
• examples: travel planning; planning of building activities
• automation only feasible, if the basic plan elements are predefined
• consider use of the general synthesis method (e.g. therapy planning) or the configuration-design method
Scheduling

- Given a set of predefined jobs, each of which consists of temporally sequenced activities called units, assign all the units to resources at time slots
  - production scheduling in plant floors
- Terminology: job, unit, resource, schedule
- Often done after planning (= specification of jobs)
- Take care: use of terms “planning” and “scheduling” differs
In applications: typical task combinations

• monitoring + diagnosis
  – Production process
• monitoring + assessment
  – Nursing task
• diagnosis + planning
  – Troubleshooting devices
• classification + planning
  – Military applications
Knowledge model construction
• so far: focus on knowledge model as product
• bottleneck for inexperienced knowledge modelers
  – how to undertake the process of model construction.
• solution: process model
  – as prescriptive as possible
  – process elements: stage, activity, guideline, technique
• but: modeling is constructive activity
  – no single correct solution nor an optimal path
• support through a number of guidelines that have proven to work well in practice.
• knowledge modeling is specialized form of requirements specification
  – general software engineering principles apply
Stages in Knowledge-Model Construction

**STAGES**

knowledge identification

- domain familiarization
  - information sources, glossary, scenarios
- list potential model components for reuse
  - task- and domain-related components

knowledge specification

- choose task template
  - provides initial task decomposition
- construct initial domain conceptualization
  - main domain information types
- complete knowledge-model specification
  - knowledge model with partial knowledge bases

knowledge refinement

- validate knowledge model
  - paper simulation, prototype of reasoning system
- knowledge-base refinement
  - complete the knowledge bases
Stage 1: Knowledge identification

• goal
  – survey the knowledge items
  – prepare them for specification

• input
  – knowledge-intensive task selected
  – main knowledge items identified.
  – application task classified
    • assessment, configuration, combination of task types

• activities
  – explore and structure the information sources
  – study the nature of the task in more detail
Exploring information sources

• Factors
  – Nature of the sources
    • well-understood?, theoretical basis?
  – Diversity of the sources
    • no single information source (e.g. textbook or manual)
    • diverse sources may be conflicting
    • multiple experts is a risk factor.

• Techniques
  – text marking in key information sources
  – some structured interviews to clarify perceived holes in domain

• main problem:
  – find balance between learning about the domain without becoming a full
Guidelines

- Talk to people in the organization who have to talk to experts but are not experts themselves
- Avoid diving into detailed, complicated theories unless the usefulness is proven
- Construct a few typical scenarios which you understand at a global level
- Never spend too much time on this activity. Two person weeks should be maximum.
Results exploration

• **Tangible**
  – Listing of domain knowledge sources, including a short characterization.
  – Summaries of selected key texts.
  – Glossary/lexicon
  – Description of scenarios developed.

• **Intangible**
  – your own understanding of the domain
    • most important result
List potential components

- goal: pave way for reusing components
- two angles on reuse:
  - Task dimension
    - check task type assigned in Task Model
    - build a list of task templates
  - Domain dimension
    - type of the domain: e.g. technical domain
    - look for standardized descriptions
      AAT for art objects ontology libraries, reference models, product model libraries
Stage 2:
Knowledge specification

- **goal**: complete specification of knowledge except for contents of domain models
  - domain models need only to contain example instances
- **activities**
  - Choose a task template.
  - Construct an initial domain conceptualization.
  - Specify the three knowledge categories
Choose task template

• baseline: strong preference for a knowledge model based on an existing application.
  – efficient, quality assurance

• selection criteria: features of application task
  – nature of the output: fault category, plan
  – nature of the inputs: kind of data available
  – nature of the system: artifact, biological system
  – constraints posed by the task environment:
    • required certainty, costs of observations.
Guidelines for template selection

- prefer templates that have been used more than once
  - empirical evidence
- construct annotated inference structure (and domain schema)
- if no template fits: question the knowledge-intensity of the task
Guidelines

- use as much as possible existing data models:
  - useful to use at least the same terminology basic constructs
  - makes future cooperation/exchange easier
- limit use of the knowledge-modeling language to concepts, sub-types and relations
  - concentrate on "data"
  - similar to building initial class model
- If no existing data models can be found, use standard SE techniques for finding concepts and relations
  - use “pruning” method
- Constructing the initial domain conceptualization should be done in parallel with the choice of the task template
  - otherwise: fake it
Complete model specification

- Route 1: Middle-out
  - Start with the inference knowledge
  - Preferred approach
  - Precondition: task template provides good approximation of inference structure.

- Route 2: Middle-in
  - Start in parallel with task decomposition and domain modeling
  - More time-consuming
  - Needed if task template is too coarse-grained
Middle-in and Middle-out

- Middle-in
- Middle

- Knowledge Base Definitions
- Domain Schema Concepts, Relations, Rule Types

Expressions in knowledge bases
• inference structure is detailed enough, if the explanation it provides is sufficiently detailed
• inference structure is detailed enough if it is easy to find for each inference a single type of domain knowledge that can act as a static role for this inference
Guidelines for specifying task knowledge

• begin with the control structure
  – "heart" of the method

• neglect details of working memory
  – design issue

• choose role names that clearly indicate role
  – "modeling is naming"

• do not include static knowledge roles

• real-time applications: consider using a different representation than pseudo code
  – but: usage of "receive"
Guidelines for specifying domain knowledge

• domain-knowledge type used as static role not required to have exactly the “right” representation
  – design issue;
  – key point: knowledge is available.
• scope of domain knowledge is typically broader than what is covered by inferences
  – requirements of communication, explanation
Stage 3: Knowledge Refinement

- Validate knowledge model
- Fill contents of knowledge bases
Fill contents of knowledge bases

- schema contains two kinds of domain types:
  - information types that have instances that are part of a case
  - knowledge types that have instances that are part of a domain model
- goal of this task: find (all) instances of the latter type
- case instances are only needed for a scenario
Guidelines for filling contents

- filling acts as a validation test of the schema
- usually not possible to define full, correct knowledge base in the first cycle
- knowledge bases need to be maintained
  - knowledge changes over time
- techniques:
  - incorporate editing facilities for KB updating, trace transcripts, structured interview, automated learning, map from existing knowledge bases
Validate knowledge model

- internally and externally
- verification = internal validation
  - “is the model right?”
- validation = validation against user requirements
  - "is it the right model?"
Validation techniques

- **Internal**
  - structured walk-throughs
  - software tools for checking the syntax and find missing parts

- **External**
  - usually more difficult and/or more comprehensive.
  - main technique: simulation
    - paper-based simulation
    - prototype system
### Paper-based simulation

<table>
<thead>
<tr>
<th>the user says: “the car does not start”</th>
<th>DIAGNOSIS: Complaint: engine-behavior.status = does-not-start</th>
<th>Complaint is received, for which a diagnostic task is started</th>
</tr>
</thead>
<tbody>
<tr>
<td>a possible cause is that the fuel tank is empty</td>
<td>COVER: hypothesis; fuel-tank.status = empty</td>
<td>One of the three possible causes is produced.</td>
</tr>
<tr>
<td>in that case we would expect the gas indicator to be low</td>
<td>PREDICT: expected-finding: gas-dial.value = zero</td>
<td>The expected finding provides us with a way of getting supporting evidence for hypothesis</td>
</tr>
</tbody>
</table>
• model development is a cyclic process
• models act as information repositories
  – continuously updated
• but: makes requirements for support tools stronger
  – transformation tools
• Knowledge model specification
• list of all information sources used.
• list of model components that we considered for reuse.
• scenarios for solving the application problem.
• results of the simulations undertaken during validation
• Elicitation material (appendices)
Summary process

- Knowledge identification
  - familiarization with the application domain
- Knowledge specification
  - detailed knowledge analysis
  - supported by reference models
- Knowledge refinement
  - completing the knowledge model
  - validating the knowledge model
- Feedback loops may be required
  - simulation in third stage may lead to changes in specification
  - Knowledge bases may require looking for additional knowledge sources.
  - general rule: feedback loops occur less frequently, if the application problem is well-understood and similar problems have been tackled
Elicitation of expertise

• Time-consuming
• Multiple forms
  – e.g. theoretical, how-to-do-it
• Multiple experts
• Heuristic nature
  – distinguish empirical from heuristic
• Managing elicitation efficiently
  – knowledge about when to use particular techniques
Expert types

• Academic
  – Regards domain as having a logical structure
  – Talks a lot
  – Emphasis on generalizations and laws
  – Feels a need to present a consistent “story”: teacher
  – Often remote from day-to-day problem solving

• Practitioner
  – Heavily into day-to-day problem solving
  – Implicit understanding of the domain
  – Emphasis on practical problems and constraints
  – Many heuristics
Human limitations and biases

- Limited memory capacity
- Context may be required for knowledge recollection
- Prior probabilities are typically under-valued
- Limited deduction capabilities
Elicitation techniques

- Interview
- Self report / protocol analysis
- Laddering
- Concept sorting
- Repertory grids
Interview
Interview: Session preparation

- Establish goal of the session
- Consider added value for expert
- Describe for yourself a profile of the expert
- List relevant questions
- Write down opening and closing statement
- Check recording equipment
  - audio recording is usually sufficient
- Make sure expert is aware of session context: goal, duration, follow-up, et cetera
Interview: Start of the session

- Introduce yourself (if required)
- Clarify goal and expectations
- Indicate how the results will be used
- Ask permission for tape recording
- Privacy issues
- Check whether the expert has some questions left
- Create as much as possible a mutual trust
• Avoid suggestive questions
• Clarify reason of question
• Phrase questions in terms of probes
  – e.g., “why …”
• Pay attention to non-verbal aspects
• Be aware of personal biases
• Give summaries at intermediate points
Interview: End of the session

- Restate goal of the session
- Ask for additional/qualifying
- Indicate what will be the next steps
- Make appointments for the next meetings
- Process interview results ASAP.
- Organize feedback round with expert
- Distribute session results
Unstructured interview

- No detailed agenda
- Few constraints
- Delivers diverse, incomplete data
- Used in early stages: feasibility study, knowledge identification
- Useful to establish a common basis with expert
  - s/he can talk freely
Structured interview

- Knowledge engineer plans and directs the session
- Takes form of provider-elicitor dialogue
- Delivers more focused expertise data
- Often used for “filling in the gaps” in the knowledge base
  - knowledge refinement phase
- Also useful at end of knowledge identification or start of knowledge specification
- Always create a transcript
Interview structure for domain-knowledge elicitation

- Identify a particular sub-task
  - should be relatively small task, e.g. an inference
- Ask expert to identify “rules” used in this task
- Take each rule, and ask when it is useful and when not
- Use fixed set of probes:
  - “Why would you do that?”
  - “How would you do that?”
  - “When would you do that?”
  - “What alternatives are there for this action?”
  - “What if …?”
  - “Can you tell me more about ..?”
Interview pitfalls

- Experts can only produce what they can verbalize
- Experts seek to justify actions in any way they can
  - “spurious justification”
- Therefore: supplement with techniques that observe expertise “in action”
  - e.g. self report
Self report
Self report

- Expert performs a task while providing a running commentary
  - expert is “thinking aloud”
- Session protocol is always transcribed
  - input for protocol analysis
- Variations:
  - shadowing: one expert performs, a second expert gives a running commentary
  - retrospection: provide a commentary after the problem-solving session
- Theoretical basis: cognitive psychology
Requirements for self-report session

• Knowledge engineer must be sufficiently acquainted with the domain
• Task selection is crucial
  – only a few problems can be tackled
  – selection typically guided by available scenario’s and templates
• Expert should not feel embarrassed
  – consider need for training session
Analyzing the self-report protocol

- Use a reference model as a coding scheme for text fragments
  - Task template
- Look out for “when”-knowledge
  - Task-control knowledge
- Annotations and mark-ups can be used for domain-knowledge acquisition
- Consider need for tool support
Self report guidelines and pitfalls

- Present problems in a realistic way
- Transcribe sessions as soon as possible
- Avoid long sessions (maximum = 20 minutes)
- Presence of knowledge engineer is important
- Be aware of scope limitations
- Verbalization may hamper performance
- Knowledge engineer may lack background knowledge to notice distinctions
Use of self reports

- Knowledge specification stage
- Validation of the selection of a particular reference model
- Refining / customizing a task template for a specific application
- If no adequate task template model is available: use for bottom-up reasoning model construction
  - but: time-consuming
Laddering
• Organizing entities in a hierarchy
• Hierarchies are meant as pre-formal structures
• Nodes can be of any type
  – class, process, relation, ….
• Useful for the initial phases of domain-knowledge structuring
  – in particular knowledge identification
• Can be done by expert
  – tool support
Example ladder

[Diagram of a ladder with attributes like role, smoker, gender, size, location, and options for secretary, manager, researcher, yes, no, M, F, single, large, central, non-central]
Concept sorting
Concept sorting

- Technique:
  - present expert with shuffled set of cards with concept names
  - expert is asked to sort cards in piles
- Helps to find relations among a set of concepts
- Useful in case of subtle dependencies
- Simple to apply
- Complementary to repertory grids
  - concept sort: nominal categories
  - repertory grid: ordinal categories
Card sort tool
Repertory grids
• Based on personal construct theory (Kelly, 1955)
• Subject: discriminate between triads of concepts
  – Mercury and Venus versus Jupiter
• Subject is asked for discriminating feature
  – E.g. “planet size”
• Re-iterate until no new features are found
• Rate all concepts with respect to all features
• Matrix is analyzed with cluster analysis
• Result: suggestions for concept relations
• Tool support is required
Example grid
When to use which technique?

- **Knowledge identification**
  - Unstructured interview, laddering

- **Knowledge specification**
  - Domain schema: concept sorting, repertory grid
  - Template selection: self report
  - Task & inference knowledge: self report

- **Knowledge refinement**
  - Structured interview
EXAMPLE
Housing application

- An application for assigning houses to potential renters
- We now sketch the organization, task and agent model and build the knowledge model on top.
Problem description

- Local government institution is responsible for assignment of rental houses to applicants
- Transparent assignment procedure
  - two-weekly magazine with house offers
  - publication of results
- Partially automated process
- Existing databases of applicants and residences
Organization models

Organization Model

OM-1

Problems & Opportunities

General Context
(Mission, Strategy, Environment, CSF’s, …)

Potential Solutions

OM-2

Organization Focus
Area Description

Structure

Process

People

Culture & Power

Resources

Knowledge

OM-3

Process Breakdown

OM-4

Knowledge Assets
### Problems and Opportunities Worksheet Organization model 1

<table>
<thead>
<tr>
<th>Organization Model</th>
<th>Problems and opportunities</th>
</tr>
</thead>
</table>
| Problems and opportunities | assessment takes too much time  
not sufficient time for urgent cases |

| Organizational context | Mission: transparency of procedure, clear applicant responsibility  
External actors: local council, public opinion, national regulations, …  
Strategy: broaden scope of market |

| Solutions | 1. Automated assessment system &  
2. Training program for assessors to be come urgency handlers |
## Organization model 2

<table>
<thead>
<tr>
<th>Organization model</th>
<th>Variant aspects: Worksheet Organization model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources</td>
<td>Existing database of applicants and residences</td>
</tr>
<tr>
<td></td>
<td><strong>Priority calculator</strong> for computing a priority list</td>
</tr>
<tr>
<td></td>
<td>of applicants for a residence.</td>
</tr>
<tr>
<td>Knowledge</td>
<td><strong>Assessment criteria</strong>: knowledge for judging</td>
</tr>
<tr>
<td></td>
<td>correctness of individual applications</td>
</tr>
<tr>
<td></td>
<td><strong>Assignment rules</strong>: knowledge used for</td>
</tr>
<tr>
<td></td>
<td>selecting an applicant for a particular house.</td>
</tr>
<tr>
<td></td>
<td><strong>Urgency rules</strong>: special rules and regulations for urgent cases (e.g., handicapped people).</td>
</tr>
<tr>
<td>Culture &amp; power</td>
<td>Hierarchical organization</td>
</tr>
<tr>
<td></td>
<td>Employees view the future with some trepidation</td>
</tr>
<tr>
<td></td>
<td>Management style: history as civil servant</td>
</tr>
<tr>
<td>Task</td>
<td>Performed by</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>1. Magazine production</td>
<td>Magazine editor</td>
</tr>
<tr>
<td>2. Data entry applications</td>
<td>Data typist / automated telephone</td>
</tr>
<tr>
<td>3. Application assessment</td>
<td>Assigner</td>
</tr>
<tr>
<td>4. Residence assignment</td>
<td>Assigner</td>
</tr>
</tbody>
</table>
Organization model 4

• Knowledge asset:
  – “general residence-application norms”
• right form?
  – no, should be also in electronic form
• right place, time, quality?
  – yes
Task model

- Task = subpart of a business process
- goal-oriented value-adding activity
- handles inputs and delivers desired outputs
  - in a structured and controlled way
- consumes resources;
- requires (and provides) knowledge/skills
- adheres to quality and performance criteria
- carried out by responsible and accountable agents
Task model

Information Systems (3D) View

Managerial View

(Task)Function and Flow

Time and Control

Knowledge and Competences

Resources

Goal and Value

Performance and Quality

Agents

Objects Structure (Data)
Task model: data flow

applicant → data entry checking → assessment → valid application → assign → assignments

Legend

- (external) actor
- processing function
- data store
- data flow
Task model: control flow

application received before deadline

- data entry checking
  - [data = correct] further processing
  - [data = incorrect] garbage bin

assessing
  - [decision = not eligible]
  - [decision = eligible]
Agent model

- OM and TM => process/task perspective
- AM: perspective of individual agents
  - staff, software systems
- large part: rearrangement of information already in other worksheets
  - just a single worksheet
- agent view useful for judging impact
  - See attitude matrix
- important input for communication model
Agent model

- **Directorate**
  - Director
  - Deputy Director

- **Policy Department**
  - Statistical Analyst Staff Member

- **Public Service**
  - Magazine Editor
  - Magazine Producer
  - Information Officer

- **Residence Assignment**
  - Data Entry Assigner

- **Computer Support**
  - Database Administrator
  - System Analyst
<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>Assigner</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Organization</strong></td>
<td>Residence-assignment department</td>
</tr>
<tr>
<td><strong>Involved In</strong></td>
<td>3. Application assessment</td>
</tr>
<tr>
<td></td>
<td>4. Residence assignment</td>
</tr>
<tr>
<td><strong>Communicates with</strong></td>
<td>Database</td>
</tr>
<tr>
<td></td>
<td>Priority calculator</td>
</tr>
<tr>
<td><strong>Knowledge</strong></td>
<td>Assessment criteria</td>
</tr>
<tr>
<td></td>
<td>Assignment rules</td>
</tr>
<tr>
<td></td>
<td>Urgency rules</td>
</tr>
<tr>
<td><strong>Other competencies</strong></td>
<td>Ability to handle problematic non-standard cases</td>
</tr>
<tr>
<td><strong>Responsibilities &amp; constraints</strong></td>
<td>Make sure that people are treated equally (no favors). This has been a problem in the past</td>
</tr>
</tbody>
</table>
Knowledge model

- Reading the two-weekly magazine in detail
  - organizational goal of transparent procedure makes life easy
- Reading the original report of the local government for setting up the house assignment procedure
  - identification of detailed information about handling urgent cases
- Short interviews/conversations
  - staff member of organization
  - two applicants (the “customers”)
- Now we look into
  - Domain model
  - Inference structure
  - Task layer
Domain model

**person**
- number: natural
- category: { starter-residence, followup-residence }
- build-type: { house, apartment }
- street-address: string
- city: string
- num-rooms: natural
- rent: number
- max-num-inhabitants: natural
- subsidy-type: subsidy-type-value
- surface-in-square-meters: natural
- floor: natural
- lift-available: boolean

**applicant**
- registration-number: string
- application-type: { starter, existing-resident }
- name: string
- street-address: string
- city: string
- birth-date: date
- age: natural
- age-category: age-category-value
- gross-yearly-income: natural
- household-size: natural
- household-type: household-type-value

**residence application**
- application-date: string

**residence criterion**
- truth-value: boolean

- correct house category
- correct household size
- rent fits income
- residence-specific constraints
Inference structure

- **Case**: raw data about a residence and a applicant e.g. age, income, rent
- **Abstract**: abstraction such as age category are added to the case
- **Abstracted Case**: case-specific norms rules that only apply to one particular residence
- **Specify**: criteria such as “rent fits income” “correct household size”
- **Select**: norm a single criterion e.g. rent fits income
- **Evaluate**: norm value e.g. rent fits income = true (or false)
- **Match**: decision applicant is either eligible or not eligible for the residence
Task layer

- **Task**
  - **Task method**
    - **Asses Case**
      - **Asses through abstract & match**
        - **Abstract Case**
        - **Match Case**
          - **Abstract Match**
          - **Match Method**

- **Inference**
  - **Abstract**
  - **Specify**
  - **Select**
  - **Evaluate**
  - **Match**
Summary

• Knowledge model components
  – Knowledge model:
    • specialized tool for specification of knowledge-intensive tasks
    • abstracts from communication aspects
    • real-world oriented
    • reuse is central theme
  – Task knowledge
    • goal-oriented
    • functional decomposition
  – Domain knowledge
    • relevant domain knowledge and information
    • static
  – Inference knowledge
    • basic reasoning steps that can be made in the domain knowledge and are applied by tasks

• Template knowledge models
  – Knowledge models partially reused in new applications
  – Type of task = main guide for reuse
  – Catalog of task templates
  – reusable combination of model elements
    • (provisional) inference structure
    • typical control structure
    • typical domain schema from task point-of-view
  – specific for a task type
  – supports top-down knowledge modeling
Knowledge model construction

**STAGES**

- **knowledge identification**
  - domain familiarization (information sources, glossary, scenarios)
  - list potential model components for reuse (task- and domain-related components)

- **knowledge specification**
  - choose task template (provides initial task decomposition)
  - construct initial domain conceptualization (main domain information types)
  - complete knowledge-model specification (knowledge model with partial knowledge bases)

- **knowledge refinement**
  - validate knowledge model (paper simulation, prototype of reasoning system)
  - knowledge-base refinement (complete knowledge bases)
Summary (cont’d)

• Knowledge elicitation techniques
  – Interview
  – Self report / protocol analysis
  – Laddering
  – Concept sorting
  – Repertory grids
  – Automated learning techniques
    • Induction
REFERENCES
• Mandatory reading:
    • Chapters 1, 2, 4, 6-8
• Further reading:
  – [http://www.commonkads.uva.nl](http://www.commonkads.uva.nl)
• Wikipedia links:
<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Propositional Logic</td>
</tr>
<tr>
<td>3</td>
<td>Predicate Logic</td>
</tr>
<tr>
<td>4</td>
<td>Reasoning</td>
</tr>
<tr>
<td>5</td>
<td>Search Methods</td>
</tr>
<tr>
<td>6</td>
<td>CommonKADS</td>
</tr>
<tr>
<td>7</td>
<td><strong>Problem-Solving Methods</strong></td>
</tr>
<tr>
<td>8</td>
<td>Planning</td>
</tr>
<tr>
<td>9</td>
<td>Software Agents</td>
</tr>
<tr>
<td>10</td>
<td>Rule Learning</td>
</tr>
<tr>
<td>11</td>
<td>Inductive Logic Programming</td>
</tr>
<tr>
<td>12</td>
<td>Formal Concept Analysis</td>
</tr>
<tr>
<td>13</td>
<td>Neural Networks</td>
</tr>
<tr>
<td>14</td>
<td>Semantic Web and Services</td>
</tr>
</tbody>
</table>
Questions?