2. Structure of a KBS: Viewpoints

There are 3 principle roles for those who work with KBS:

1. **End user**
   Doctor, factory manager, person of the street, etc.

2. **Knowledge engineer**
   The person who designs, classifies, organises, builds a representation of ... the knowledge)

3. **Constructor of the tools for the KBS**
   Typically a programmer

   - Often, the last 2 roles are held by the same person (less so on big projects).
   - Descriptions of the KBS differ for each role.
2.1. The End-user’s View

• The end user sees the system basically as a black box to solve their problems.

• They thus see the system as a user interface (with which they interact), and some intelligent program behind it, which does its job using some unknown logic.

• They do not usually distinguish the knowledge base (domain knowledge) from the inference engine which uses this knowledge.
2.1. The End-user’s View (ii)

Intelligent Program
• For the user, a black-box that follows some unknown logic but produces the results the user desires

User Interface (UI)
• Human-computer interaction is of greatest importance for the user: menus, (pseudo-) natural language, graphical and visual elements.
• The UI might:
  – Allow the user to specify the problem,
    • Possibly graphically, via form-filling, or even natural language dialog
  – Present solution of the problem
  – Present user with the reasoning used to derive its solution
• This is a very important component of any KBS
  – a whole area of informatics is devoted to it: Human-Computer Interaction
  – A good UI can make a poor KBS look good, and visa versa

2.1. The End-user’s View (ii)

Problem-specific Database
• Can contain:
  – Data input by the user
  – Conclusions (both intermediate and final) of the system
• Facts (can be uncertain, as with knowledge)
• Facts refer to a specific instance of the problem (a patient entering a hospital, a credit application ... )
• End-user is responsible for the facts
2.2. The Knowledge Engineer's View

The job of the Knowledge Engineer is to collect the expert knowledge needed for an expert system, and enter it into the system.

They may extract knowledge from human experts, or from data from other sources (e.g., sensors, databases, etc.)

As managers of the Knowledge Base, they are aware of the separation between the inference engine and the knowledge it uses.

The KE’s view of the system is shaped by the tools they use to interact with knowledge

---

2.2. The Knowledge Engineer's View

The Tasks of a Knowledge Engineer (i):
Knowledge Acquisition/Management

The main task of the KE is knowledge entry and maintenance. For this, a special user interface is provided:

- **Knowledge Acquisition Tool**: a tool for entering/maintaining knowledge into the knowledge base.

  The KAT might provide other functionalities:
  - consistency checking (of the KB)
  - bookkeeping functions (who edited what and when)
  - access control (view vs. edit permissions; changes by lower level KEs need to be approved by higher level KEs, etc.)
The Knowledge Engineer’s view

2.2. The Knowledge Engineer’s View

The Tasks of a Knowledge Engineer (ii): System testing

- The second task of the KE is to test that the system as a whole works after some modification to the knowledge-base, or after a change to the inference engine (extension of power, change of algorithm, etc.)
- For this, the KE might use the following:
  - **Developer’s Interface**: basically the same as the end-user interface, but offering a more detailed view of the inside:
    - Run-time trace of reasoning steps,
    - Access to inner components of the system,
    - etc.
  - **Test Case Database**: After adding, deleting or modifying the knowledge system, the reasoning of the system may be affected. To allow the KE to test the system, a database of sample problems, called the test case database, is usually developed, for testing the system.
    After a substantial change, the KE will run through the test cases to verify the system is working as expected.
2.2. The Knowledge Engineer's view

- The two interfaces for the KE can sometimes be grouped together as a "development shell", a general KE Workbench.
2.3. The Tool Builder’s View

- The Tool Builder is concerned with building an adequate set of tools for the Knowledge Engineer to use to build the Knowledge base.
- They thus see the system from the same viewpoint as the KE, except with a different focus.
- More concerned with implementation decisions:
  - knowledge representation (rules, frames, semantic networks, ...)
  - inference mechanisms (deduction, uncertainty, ...)
- The tool builder is also concerned to support the final user.
- They are thus concerned with:
  - user-friendliness of the user interface,
  - the kinds of functionalities supported (what reports, views, etc.)

2.4. Knowledge Based Systems

- Development tool: A suite of software that allows construction of a knowledge base and interaction with this knowledge base through use of an inference engine.
- May be used by Knowledge Engineer (to develop the KB and test it) or the Final User (with a KB).
- Development tools can be acquired in 2 ways:
  - Developed from scratch: KBS Tools may be developed for a particular project from scratch.
  - Obtain an “off the shelf” system: a system which comes with no knowledge base of its own, but will work with a user-provided knowledge base. Often called an “expert system shell”
2.4. Expert System Shells

Expert System Shell: Example
• Originally, MYCIN was developed for a single application: medical diagnosis
• The developers had designed the system to properly separate knowledge from the inference engine.
• They thus released a version called E-MYCIN (‘E’ for ‘empty’), which allowed users to provide their own knowledge base.
• As long as their application used the same types of inference, the system should work for them in the new domain.
• E-MYCIN functionality:
  – reasoning via backward chaining
  – management of uncertainty
  – could generate explanations
  – a user interface

2.4. Expert System Shells

“Do it yourself” systems:
• Avoiding shells, there is the “do it yourself” approach

Advantages:
  – cheap (in purchasing price)
  – can customise to your particular problem
  – In the end, company owns the system

Disadvantages:
  – more expensive (in person months)
  – more risk due to unknowns