Chapter Overview

Expert System Development

Development Issues

Models

Rapid Prototyping and Incremental Development

Knowledge Engineering Lifecycle

Linear Model

Error Sources

Problem Selection

problem identification
What exactly is the problem to be solved?

users
Who is going to use the system?

expertise
Where does the knowledge come from?

appropriateness
Is an expert system the right tool?

tools
Are tools available for building the system?

payoff
Time savings, better efficiency, better products, , ...

cost
Hardware, software, training, people, ...

Development Issues

project management

- activity management
  planning, scheduling, monitoring, analysis
- product configuration
  versions, changes
- resources
  forecast and acquire resources
  assign responsibilities
  critical resources for bottlenecks

system delivery
standard hardware / OS
integration with existing systems

maintenance and evolution
system may continually evolve.

Development Stages

feasability study
Can it be done?

rapid prototype
quick implementation to give an impression of the overall system

refined system α - test
in-house test on real problems

field testable β - test
tests by selected users (non-specialists)

commercial quality system
validated and tested
documentation, training, support

maintenance / evolution
bugs fixed
capabilities enhanced

Conventional Software Lifecycle

experience
- much longer history than expert systems
- much larger base of realized systems

methodologies
- variety of models to describe the software development process

development tools
- reasonable choice of proven tools

Waterfall Model

traditional, widely used

problem analysis
- suitability of the problem
- costs and benefits
- potential users

requirements specification
- formal document
- goals and features of the system
- expected users
- computational environment
- constraints

design
- choice of tools (software, hardware)
- user interface
- system architecture
- design documents

implementation
- writing and debugging code
- integration of modules
- interface to external components / systems

testing
- specifications must be met
- proper solution to the problem
- correct operation
- simulation or real environment

maintenance
- elimination of errors
- modifications (e.g. for improved performance)
- enhancements
- most costly of the lifecycle stages

Waterfall Model: Advantages and Problems

+ clear methodology

+ methodical approach

+ stepwise realization

\- serial nature

\- requires deep design knowledge from the early stages

\- user feedback only at the very end

\- long time between project conception and implementation

Boehm Spiral Model

combines waterfall model, prototyping, and risk analysis

cyclic repetition of steps

radial dimension: accumulated costs

angular dimension: progress in a phase

steps in a cycle

- identification
  objectives, alternatives, constraints
- evaluation
  examines the previously identified issues
- formulation
  of a strategy to solve uncertainties and risks
- assessment
  of remaining risks
  progress to the next step / component

Boehm Spiral Model: Advantages and Problems

\+ realistic view for large-scale software system development

\+ stepwise approach

\+ incremental realization

\+ explicit risk assessment

\- more complex

\- evolutionary process

\- heavy reliance on risk assessment

Differences

Software Lifecycle – Knowledge-Based System Lifecycle

software
algorithms
data structures

knowledge-based system
heuristics
structured knowledge

Rapid Prototyping

working prototype
quick creation of a limited version of the envisioned system

feasibility
demonstrates that the system can be built

design issues
evaluation of basic design choices

design changes
can be made early in the development phase

customer feedback
early integration of requests

Incremental Development

divide-and conquer
concentrates on manageable, separate chunks of knowledge

iterative development
the chunks of knowledge are elicited from the source of expertise, implemented, reviewed, and refined

permits parallel development

Knowledge Engineering Lifecycle

from initial model to retirement

problem analysis
nature of the problem
potential users
available resources
adequacy of expert system methods
costs and benefits

requirements specification
formalization of the problem analysis results
objectives of the project
means to obtain the objectives

preliminary design
high-level design decisions

- knowledge representation method
- development tools
- sources of expertise

foundation for the initial prototype

initial prototyping
looks like the complete system
limited in breadth
used to justify or overturne preliminary design decisions
usually discarded

final design
high level description of the system architecture
identification of subsystems
interfaces between subsystems
selection of

- tools
- resources
- knowledge representation method

implementation
complete knowledge acquisition
incremental development

validation and verification "V & V"
ensures that the system meets its specification
tests the operation of the system

design adjustment
significant or retroactive changes may result in a paradigm shift

maintenance and evolution
elimination of bugs
adaptation to user requests
integration of new or modified knowledge
enhancement of functionality

Linear Model

for expert systems development [1]

planning

- feasibility assessment
- appropriateness of expert system methods
- resource management
- task phasing
- schedules
- preliminary functional layout
- high-level requirements

results in a formal set of documents (work plan)

knowledge definition

knowledge source identification and selection

[1] Chapter 6 in [Giarratano and Riley, 1994]; originally developed by [Bochsler, 1988]

- source identification
- source importance
- source availability
- source selection

knowledge acquisition, analysis and extraction

- acquisition strategy
  methods, access to sources
- knowledge element identification
  select useful knowledge items, sources
- knowledge classification system
  organization of the knowledge
  (hierarchical groups)
- detailed functional layout
  functional capabilities at a technical level
- preliminary control flow
  general phases during execution
  groups of rules
- preliminary user's manual
  to elicit early feedback

- requirements specification
- knowledge baseline
  formal basis for changes (change requests)

knowledge design

knowledge definition

- knowledge representation
  rules, frames, logic
- detailed control structure grouping of rules
  interface with other components
  metalevel control structures
- internal fact structure
  e.g. `deftemplate`
- preliminary user interface
- initial test plan
  test data, test drivers
  analysis of test results

detailed design

- design structure
  logical organization of knowledge
- implementation strategy
- detailed user interface
  after user feedback about the preliminary version
- design specifications
  formal document
- detailed test plan

code and checkout

actual code implementation

- coding
- tests
- source listings
  commented, with documentation
- user manual
- installation / operations guide

- system description
  formal document

terminates with the test readiness review

knowledge verification
formal tests

- test procedures
- test reports

test analysis

- results evaluations
- recommendation

system evaluation

- summary results evaluation
- recommendations
- validation
  system fulfills requirements and operates correctly

- final report (complete system)
  interim report if modifications need to be made

final stage in the development
refinements or modifications start the overall process from scratch

Advantages and Problems

+ verification and validation in parallel with stages
+ suited for large, commercial-quality expert systems
+ stepwise realization

- serial nature
- substantial overhead
- user feedback only at the very end
- long time between project conception and implementation

Error Sources

knowledge errors
the acquired knowledge may be erroneous
explicit representation of the knowledge may already uncover errors
special efforts for mission-critical projects (review panels, formal verification)

semantic errors
mis-interpretations of expert knowledge
incomplete elicitation of knowledge

syntax errors
incorrect forms for rules or facts
should be detected by the development tools

inference engine errors
bugs in the expert system's inference engine
mostly obscure, infrequent, not consistent
possible sources:

- pattern matching

- conflict resolution
- execution of actions

inference chain errors
possibly caused by combinations of above errors
priority problems with rules
side-effects between rules
uncertainty, especially propagation
nonmonotonicity

limits of ignorance
performance should degrade gracefully when the limits of knowledge are reached
ignorance should increase uncertainty
Problem: How does the system know its limits?

Chapter Review

Expert System Development

Development Issues
problem selection, management, stages

Models
conventional software vs. knowledge-based systems
waterfall, Boehm spiral, linear

Rapid Prototyping and Incremental Development
quick demonstration, subdivision, parallel work

Knowledge Engineering Lifecycle
analysis, specification, prototype, design, implementation, validation and verification, design adjustment, maintenance and evolution

Linear Model
planning, definition, design, code and checkout, verification, evaluation

Error Sources
knowledge, semantic, syntax, inference, limits of ignorance