Generating Knowledge-Based System Generators: A Software Engineering Approach

Sabine Moisan, INRIA, France

ABSTRACT

This article investigates software engineering techniques for designing and reengineering knowledge-based system generators, focusing on inference engines and domain specific languages. Indeed, software development of knowledge-based systems is a difficult task. We choose a software engineering approach to favor code reuse, evolution, and maintenance. We propose a software platform named Lam\a to design the different elements necessary to produce a knowledge-based system. This platform offers software toolkits (mainly component frameworks) to build interfaces, inference engines, and expert languages. We have used the platform to build several KBS generators for various tasks (planning, classification, model calibration) in different domains. The approach appears well fitted to knowledge-based system generators; it allows developers a significant gain in time, as well as it improves software readability and safeness.

Keywords: Component Frameworks, Generators of Knowledge-Based Systems, Reuse, Software Engineering

SOFTWARE ENGINEERING AND ARTIFICIAL INTELLIGENCE DEVELOPMENT

Among the different types of cooperation between artificial intelligence (AI) and software engineering, many deal with applying AI techniques for better software development. By contrast, we propose to use software engineering techniques to improve AI software development, in particular knowledge-based system design.

Knowledge-Based Systems (KBS) are programs that rely on explicit knowledge to automatically perform various tasks usually devoted to human beings. A task here means an application-independent problem-solving activity, such as classification, diagnosis, design, or planning. Indeed, creating a new KBS relies on a model of the task to perform. Such a model includes two major categories of knowledge. First, a reasoning strategy, often named problem solving method, mirrors the reasoning process of an expert performing the task. Second, concepts make explicit the important features involved in the task and in the application domain; they are usually modeled by means of ontologies.

Knowledge-based systems are mainly composed of three parts: a knowledge base storing expertise in a particular domain, a fact base
containing facts about an end-user problem in
this domain, and an engine, written by a software
designer. Relying on the expert’s knowledge,
the engine performs inferences to solve the end-
user problem. All actors (designer, expert, and
designer) should have tools that correspond to
their needs. A lot of work have been devoted
in the knowledge engineering community aim
to help experts manage knowledge base devel-
opment and maintenance, through customized
high level tools and methods. However, when
developing knowledge-based systems, there
is another major source of difficulties: (re)
designing the various software tools that will
be used by experts or end-users to interact with
the system, such as inference engine, knowledge
acquisition tools, knowledge editors/verifiers,
etc. Each element by itself represents a great
amount of code. Moreover, all elements must
work together, although every one may evolve
independently. It is the designer’s job to develop,
maintain, and customize these tools. This cor-
responds to a software engineering activity
which implies to convert a cognitive model, as
expressed by experts, into a software model and
eventually an operational system implemented
in a regular programming language.

At the present time designers seldom re-
ceive adapted support. The tools they have to
cope with are usually not at the right abstrac-
tion level. Knowledge engineering tools do
not offer development, integration and testing
facilities. For instance, modifying (part of) the
behavior of an engine is difficult and prone to
tools since the semantics is often hidden inside
the engine code and mixed with programming
idiosyncrasies, that lie at a much lower level of
abstraction. In the realm of Software Engineer-
ing many works have been done on software
reuse, maintenance, evolution, and safety. We
expect that customizing such approaches to fit
KBS needs may lead to dramatic improvement
in their development and adaptation.

A META GENERATOR
APPROACH

Knowledge-based systems were first de-
veloped using ad hoc customized approaches.
Then, based on similarities among activities
and across application domains, the notion of
KBS generators emerged in the late 80s. KBS
generators take advantage of the cross-domain
similarities by abstracting the common artificial
intelligence concepts and by pooling represen-
tation techniques in the same environment.
Such development environments provide a
panel of reasoning mechanisms and knowledge
representations that can be shared and adapted
to different domains. They usually propose
knowledge acquisition, learning, and verifica-
tion tools, various editors, and an inference
engine. Some of these elements are used only
during the KBS elaboration (such as knowledge
verification tools), whereas other can be active
even during operational use of the final system
(engine or learning tools). However, all of them
are necessary to conveniently design KBSs.
Except for general rule-based shells (such as
Jess; Friedman-Hill, 2002), most generators
are more or less dedicated to a given range of
applications or to a given task (e.g., classifi-
cation) through their inference mechanisms.
They are yet domain-independent, hence their
KBS instances may apply to various domains
(e.g., classification of cardiologic diseases, of
astronomic objects, of biological organisms).
Such specialized generators are closer to expert
ways of reasoning and often lead to more ef-
ficient KBSs.

Generators have proved useful to improve
the KBS development cycle by sharing com-
mon elements. They properly meet experts’ or
end-users’ modification needs at the cognitive
level, since they support modification and
maintenance of knowledge base contents and, in
some respect, minor modifications in reasoning

Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global
is prohibited.
A Modified Fuzzy Hierarchical TOPSIS Model for Hotel Website Evaluation
[www.igi-global.com/article/a-modified-fuzzy-hierarchical-topsis-model-for-hotel-website-evaluation/94621?camid=4v1a](www.igi-global.com/article/a-modified-fuzzy-hierarchical-topsis-model-for-hotel-website-evaluation/94621?camid=4v1a)