Semantic networks for intelligent task solvers

Elena A. Shalfeyeva

Abstract— This paper is devoted to modeling of reusable components for knowledge-based systems. It is suggested that development models of program units as well as the knowledge and the database are presented as semantic networks. It offers a structure of semantic networks for models of reusable units of the constructed solvers.

Index Terms: knowledge-based system, task solver, program unit, reusability, maintainability, hierarchical semantic network

I. INTRODUCTION

Knowledge-based systems (KBS) and their maintainability still rank high in the field of development of program systems [1]. They distinguish such independent subsystems of KBS as a specialized solver of tasks, an user interface, a knowledge base, databases, an editing tools for this bases.

Transition to declarative representation of knowledge in KBS has already made an essential contribution to maintainability of knowledge bases [1, 2]. The knowledge base becomes conceptual and clear for the expert maintaining it.

In one of the modern approaches to creation of maintainable KBS the specialized task solver is provided. It is the problem-oriented program which solves an intellectual problem or supports any specialist in decision-making using the knowledge base. Since the advantages of declarative presentation of programs are well-known [3], the approach to realization of declaratively presented specialized task solver is suggested and is being evolved. One of the solver's declarative models specifies which program units carry out its functionality and what are the control communications among them [4].

The "declarative" approach mentioned above is focused on the common principles of formation, modifying and usage of the declarative descriptions of system components and the information resources (IR) processed by the components [4].

While creating such KBS there are additional requirements to program units that form the solvers. Unification of interfaces of these units, their reusability and accessibility to knowledge and data bases as semantic networks are important [5, 6].

The purpose of this research is to develop a solver's program unit "elaboration model" as semantic networks. It is a set of essential development models, each model structure and identification of relations among them. Such network model becomes a basis for of developer’s instruments on the IACPaaS platform [5].

II. SUBSYSTEMS MODELS AND THEIR PRESENTATION

Before KBS can be engineered, the "system" in which it resides must be understood. To accomplish this the overall objective of the system must be determined; roles of all system elements must be identified and operational requirements must be elicited, analyzed, specified, modeled [7]. One of the results of system modeling is system architecture flow diagram [7]. Usually its subsystems are quite complex and are composed of smaller "units" by architectural designing. Depending on a realization method, program units can take form of procedures/functions, classes, actors, agents and other units.

When such components (structural units of a subsystem) are created to be reusable special attention for their engineering is required.

Specifications for program units development are created taking into account the analysis of requirements to containing subsystem and its architectural design. According to each specification developer in charge creates full configuration model of unit (elaboration model), including a detailed project, a source code and sets of tests.

An example of requirements analysis model to a subsystem (i.e. specialized task solver) is a model of functional decomposition and/or an algorithm of a task solution. A hierarchy of subtasks can be considered as universal representation for this algorithm or functional decomposition. The subtask of some level (consecutive or with - branching) distributes into some various smaller subtasks (subfunctions), and cyclic addressing to some of them is provided.

During subsystem architectural designing all its "simulated" functionality is distributed among program units, then interaction among them is identified [7]. The quality of realization, system maintainability and their componential reusability depend on architectural models [7, 8]. Specialized solver may be constructed both by means of traditional architecture [7], and on the basis of “declarative-actors” approach [5, 6].

The Hierarchical Semantic Network is an appropriate means of representation (and storage) of those models.
The Hierarchical Semantic Network (HSN) is a network with the single root vertex and a set of nodes with in-degree equal-to-or-greater-than unity.

For the HSN the existence of two levels: meta-information and target information is typical. The level of meta-information prescribes a format of description for some information or a model.

For example, the format of representation of hierarchy of subtasks (i.e. a metainformation) can be the following:

Task {
  name: string;
  structuredness {
    ~ alt (sequential, with-branches)
  }
  ~set subtask {
    Subordination
    ~ alt (in-simple-subjection, in-cyclic-subjection, chosen-by-condition);
    -> name;
    -> structuredness;
    ~setmm { -> subtask }
  }
}

Hereinafter alternative values of the node in the meta-information, a nonempty set and a any? (random) set of elements are marked with qualifiers "alt", "set" and "setmm". The arrow before the node means a reference to the concept defined earlier.

III. MODELS OF PROGRAM UNITS ACCESSING KNOWLEDGE AND DATA IN THE FORM OF NETWORKS

In "declarative-actor" approach to KBS construction the model of functional decomposition of a task that is being solved is taken into account. Sub-functions are associated to actors - program units that will communicate through messages. Some actors will access the stored knowledge or data. Specifications and designs of each solver unit reflect their indicated peculiarities. According to the concept of "declarative" approach [5, 6] all models are formed as HSN.

When specifying the actor its “responsibility”, i.e. the result which is expected from it, is precisely indicated. This result may be returned by message from this agent. This result can be written into output IR of the solver. In some cases it can be transferred via the user interface.

The format (network structure, meta-information) of representation of actor specification may contain:

actor name: string;
appointment description: string;
~set block-function {
  block-function-description: string;
  message {
    message template: list of templates,
    parameters list: ~setmm parameter
  }
}

This meta-information is used for specification of actors on the IACPaaS platform (as we can see on fig. 1).

The actor specification, created by this meta-information, records a role of the agent in the solver or other subsystem. And the model becomes "a starting point" for designing the actor.

The design model of the actor is, in fact, a design of interfaces of the actor. Its creation is based on the specification of the actor. And in addition design model specifies information about a way of access to processed data. If a designer decides that the actor function is complex or that the actor function includes any frequently used subfunction, he may create the additional actor (co-operator) for execution of this part of work. Then in the design model of the specified actor the designer indicates all interfaces with this co-operating actor. And there will be one more output message to the new co-operating actor in the design model. Besides, it is required
that he adds the second block-function into the "specified" actor for receiving feedback messages. The second block is created for reaction to the message from the co-operating actor.

The format of design model representation may be as following:

```
Actor design {
    actor name: string;
    appointment description: string;
    block-function-description: string;
    message {
        message template: list of templates,
        parameters list: ~setmm parameter
        {parameter name: string;
        parameter type:
        ~ alt (integer, real, string, IR-pointer,...)}
    }
    ~setmm output-message {~ message};
    ~setmm input-IR:
    access-IR {parameter-IR: IR-pointer,
    meta-IR-name: IR-pointer,
    ~ set IR-operation:
    operation signature},
    ~setmm interactive information element: string;
    ~setmm output-IR: access-IR;
}
```

The content of one more important model – the ADT (abstract datatype) is also considered when constructing this model. IR ADT (also in the HSN form) is IR structure description and a set of its access operations. (IR ADT can be created in advance, for example, for the other application that are processing IR of the same format.) Each operation in the design model has to belong to the set of access operations.

At runtime on the IACPaaS platform the actor design model is used as a declarative model (for activation of the demanded block-function) [5].

When creating the reusable components, their testing completeness is particularly important. For automation of the process of test situations creation and sets of tests creation and for automation of the testing process it is necessary to store test sets in a suitable format. The network structure of its representation may be as following.

```
Actor test set {
    actor name: string;
    test {input-message {
        message content {
            message template: list of templates,
            ~set {parameter value:
            ~ alt (integer, real,
            string, IR-pointer,...
            )
        }
    }
    ~setmm input-IR {
        IR-name: IR-pointer,
        IR-state: IR-pointer,
        ~ setmm modified-IR {
        IR-name: IR-pointer,
        IR-state: IR-pointer,
        IR-expected-final-state: IR-pointer
    }
    }
```

The content of a tests set for a separate actor testing is formed on the basis of the design of its interfaces. The set of tests has to include tests for each of the implementable blocks. As we can see in the model, the input and output data for any test can be not only the content of the modified-IR and the feedback messages but also the contents of the messages that are sent to co-operating actors.

All of these models, created as HSN, are information resources that are accessed by developers and maintainers. Tools for creating of these models that are being developed on the IACPaaS platform are aimed to reduce the routine job of developers and maintainers. Besides, the tools will control consistency and completeness of the created actor model.

IV. AN EXAMPLE OF THE DEVELOPMENT MODELS OF A REUSABLE UNIT

For a medical diagnostics solver the actor "verify-hypothesis-Is-healthy" has to check the normality of all the patient’s signs.

For this purpose it can address to actor "checkNormOfSign" for each of the signs in the clinical record of the patient. After all signs are verified it will be possible to conclude whether the patient is healthy or not.

The actor "checkNormOfSign" receives two parameters (by a message) – sign’s name and the measured sign’s values. It has to organize a check of normality for the values. It needs an operation "take-normal-values" (for the indicated sign) to access IR "The Base of Normal Values".

Models of the actor are the hierarchical semantic networks edited by tools of the IACPaaS platform presented on Fig. 1 and Fig. 2.

The actor design of interfaces (see Fig. 2) is based on the specification. It specifies information about a way of access to processed knowledge base. It doesn’t have any interfaces with co-operating actors.

In the set of tests for the actor there will be tests that cover test situations for the names of the signs and for the sign’s values and test situations for the result ("normal", "abnormal"). Contents of IR "Base of Normal Values" will also be the input data for the tests.
The semantic networks describing structure for creation, storage and maintenance of the models for reusable units – actors are presented.

The suggested unit models are a basis for development and improvement of technology of construction the reusable components for KBS. The IACPaaS platform supports the controlled access and uniform administrative system for creation and usage of all KBS components and means their representation by semantic networks.

REFERENCES


