Objected-Oriented Analysis in the Application of Simulative Transformer Substation System

Cuiru Wang Jiang Jiang

Department of Computer Science, North China Electric Power University, Baoding 071003, P. R. China

Abstract

As more challenging applications are automated, solving cooperative problem will be an important paradigm for the next generation of industrial intelligent systems. One of the key problems to use it in engineering domain is development of a structured design method. In this paper, an evolutionary, seamless, non-domain-specific object-oriented analysis (OOA) method is derived that starts at the definition of a software system and integrated knowledge engineering needs in a new manner, especially when following a deep knowledge approach. Based on this proposed method, an expert-supported OOA tool environment (ESA) is presented that supports an analyst starting at the collection of the requirements through the analysis of any software system. And an example of simulative transformer substation system (STSS) is introduced to present some key problems and techniques of OOA up to a preliminary high-level design.

Keywords: Transformer substation system, Objected-oriented analysis, Expert-supported, Knowledge engineering

1. Introduction

Object orientation is not only a programming technique but a much more sophisticated approach towards engineering in general, and both software engineering and knowledge engineering in particular.

The main advantages of object-oriented technology are the universal homogeneity and the reusability of its results. The term "universal homogeneity" denotes that object-oriented technology is a much more general and universal approach to software development than the conventional method of software development. It turns out that the single steps in software development move much closer together with much more interaction among them. Thus the necessary feedback is established much earlier, and a more homogeneous and seamless development without error-prone gaps after each step is possible. Compared to conventional software development, less transformation of results is needed to get a suitable input for the next step, and the development process is much more evolutionary [1].

The reusability of the results of object-oriented technology means using what already exists to achieve what is desired. Reuse is not limited to algorithms, libraries or interface specifications but is enabled on entire applications, application frameworks and objects. Reuse at object level occurs in two different forms: refining an object via inheritance to obtain a new object and using existing objects in the composition of a new object [2].

2. Design method of ESA system based on software engineering

2.1. The main part of the OOA

Most of current application systems are very large and complex. Such a complex system's design is iterative and improved process. Summarily, the main part of the OOA system can be showed in Figure1.

When considering the entire method, it must be kept in mind that this is not a straightforward process but an iterative one with many loops and repetitions of single or multiple steps. Nor is it a sequential process in the sense that step n must be executed before step $n+1$. However, it is the basic process towards an evolutionarily developed object-oriented analysis. It is also very useful to discuss preliminary results with the expert of the domain [3] and to integrate this new feedback at the next iteration.

The whole process results in a conceptual model, a specification of the system integrating the static, dynamic and epistemic knowledge of the analyzed domain. In the case of software engineering, this output is a direct input (without any transformation) to the phase of software design, which must not necessarily follow an object-oriented approach, although it is quite recommendable, as it already anticipates a great portion of an object-oriented high-level design. However in the case, of knowledge

engineering, the output can also serve as a profound basis containing all the relevant information of the analyzed domain. It includes the elements of the domain, its components, its behavior and the relationships among the elements; it is the necessary input of an expert system.

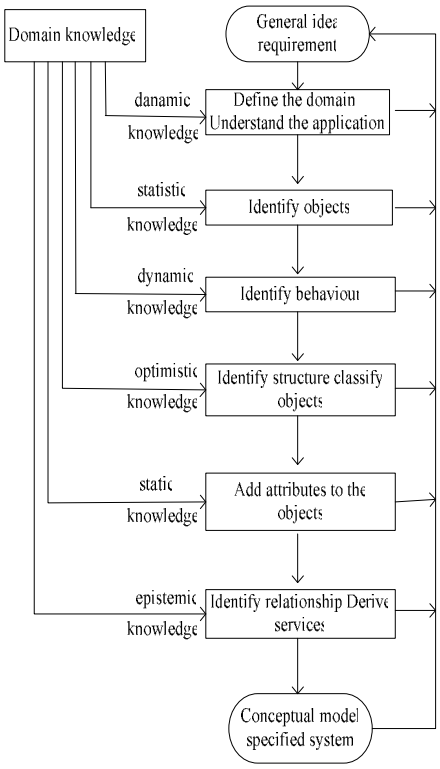

Fig.1: process of new OOA method.

The advantages of the presented method, which aim at the shortcomings of existing OOA methods and considers knowledge engineering needs, consist of the following points:

- Integration of all essential object-oriented features
- Incorporation of behaviour in an extensive way
- Support of an entire tool environment because methods are more valuable for practitioners
- Homogeneous transition to OOD and OOP
- Participation of users and domain experts during the entire process, as it is a rather pragmatic and intuitive approach
- Integration of knowledge engineering aspects

2.2. Using the salient features of ESA at an OOA process

After the general preparations for an analysis are executed, the domain of discourse is defined, and the application is understood on a high level, ESA offers a very effective way of beginning. Starting with a simple textual requirements definition, which is usually available in an informal, natural, language and is saved in a file, ESA can generate a text analysis report including a list of candidates for objects, attributes and methods.

When working without an electronic dictionary, which can identify nouns, verbs, etc., candidates for objects, methods and attributes should be identified by style attributes (bold, italic or outline). Studies have shown, however, that it is better to select the candidates manually than with an electronic dictionary. First, a better understanding of the problem is gained and, second, as the use of a dictionary does not include or consider any semantics, all possible candidates are selected without any preselection, resulting in an immense number of candidates.

When selecting any object candidate, an object will be created on the OOA workbench, or when an attribute or method candidate is selected, it will be added to the currently activated object, which is displayed highlighted. However, this can be done during the entire analysis process and can also be executed for parts of a textual description. This idea is based on the concerns described in [4], and should help to structure any badly defined domain of the particular problem.

Of course, it is also possible to create new objects by using the object creation tool and to specify their behaviour and to identify and structure their components. In order to take advantage of and to reuse already analyzed and specified objects, ESA offers the possibility to include objects already defined in an object repository and to import objects from source text (e.g., an application framework or a former project that is stored using an object-based or object-oriented language). Thus this feature essentially improves the reuse of objects and object hierarchies.

After these initial steps the next process will take place. An analyst has to identify behaviour by filling in the lower left sub window, classify objects, and identify structure by using the inheritance tool. To reduce complexity, it is advisable to create graphical super objects [5].

When the process of analysis is almost finished, the final component, the expert supporting component of ESA should be activated to analyze and criticize the model. The analysis investigates all components of an object like a compiler. It checks whether all object names are unique, whether all attribute names are defined and whether all attribute interfaces, all constraints and all behaviours are specified correctly. And the more expert-based part checks the structure of the attributes and methods within the object hierarchy. If a disadvantageous arrangement is detected, warnings and hints to rearrange it will be generated.

Also, the magnitude and the consistency of an object will be analyzed in a heuristic way to produce necessary warnings.

3. An example of simulative transformer substation

Transformer substation is an important department of electric power system. The operators of such system must be capable of managing the system both under normal conditions and in the presence of system malfunctions. Their ability to diagnose faults and take appropriate corrective actions promptly is highly desirable. But the complex and dangerous features of a transformer substation make it impossible to train operators on real equipment. Therefore developing simulative transformer substation is an effective way for training operator. It simulates the electrical network, user interface, and power system behaviour. The operator's skill will be enhanced with the sophisticated training environment. It may also be used for operator evaluations, engineering studies, power system model evaluation, and offline testing of energy management system functions and operational procedures.

Computer-based training systems for operators and dispatches have witnessed steady progress since the early 1980s [6]. Building such a system deals with a set of complex tasks, such as the control of input and output devices, load flow calculation, diagnosis of faults and abnormal events, and so on. Therefore an integrated and distributed problem solving architecture seems to be a good choice.

(1) Requirement Analysis

The aim to develop simulative transformer substation is to improve operator's operator's skill and his faults' and abnormal events' diagnosis ability. Generally, the system should include the following functions:

- to collect switch status and supervise network's change;
- to identify and diagnose faults and abnormal events;
- to calculate load flow under both fault and normal conditions;
- to display the electric network status through computer graphic interface and meters in real-time;
- to have a friendly user interface so that instructor can set up fault and abnormal events;
- to simulate various of fault and abnormal phenomena;

(2) System Overall Composition

STSS is a large complex system which combines physical simulation with computer digital simulation. It has the same look and feel as the real-time system, from both the user interface and system response perspectives. Therefore system model is based on not only its overall performance, but also the environment constrains. For example, input and output of the system should reflect their change of status logic, that is, system must be of ability of real-time input and output. In addition, in order to facilitate the instructor to set up faults and abnormal events, there must be a user interface agent. It presents the instructor with a graphical display representing the electric network status. It is mouse-driven and uses pull-down menus for ease of use [7].

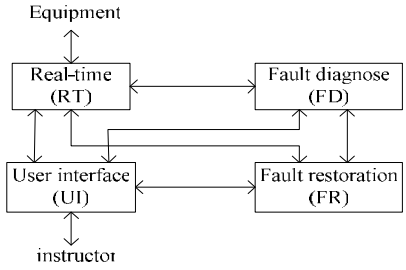

Fig.2: overall composition of STSS.

The fault diagnosis and fault restoration tasks which consume a lot of time should be performed by special agents. Therefore the system is composed of four main agents: real-time processing (RT), user interface(UI), fault diagnosis(FD), and fault restoration(FR). Figure2 shows the overall composition of STSS.

(3) Agent Model

According to the STSS's characters, we propose an agent model (see Figure 3), which has three layers They are interaction layer, cooperation and control layer, and problem solving layer. Each layer has several modules with different functions.

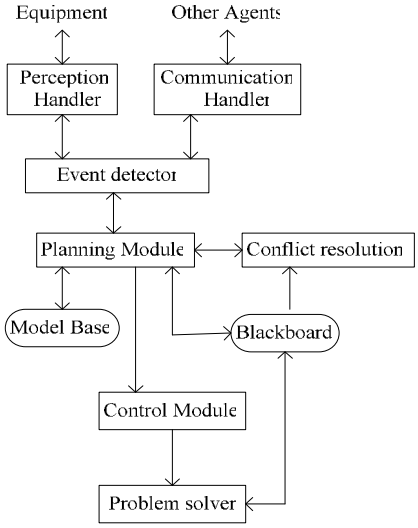

Fig.3: Architecture of an agent.

a. Interaction layer

The interaction layer is composed of perception handler and communication handler. The former is the agent's interface to user or environment, and the latter is the interface to other agents. Agent uses the perception handler to interact with environment or user such as collecting electric equipment status or receiving instruction from instructor. Communication handler is used to receive message from or send message to other agents.

b. Cooperation and control layer

The cooperation and control layer is composed of four main modules, a blackboard and a model base.

The model base includes the models of other agents and itself, which is the main information source for agent to decide what tasks should be performed locally, determine when social activity is appropriate, and whom it will interact with, and so on.

The blackboard is a shared memory for domain problem solvers and the modules in cooperation and control layer. It includes information about agent's local problem solutions, status, plan, goal, etc.

The event detector is an independent computation process and normally runs as a background process. It is triggered by the messages from either the interaction layer or the planning model. It is also responsible for coding and decoding the messages.

The planning model is agent's kernel. It is responsible for translating the abstract goals into a sequence of concrete goals and planning agent's local and social activities, such as cooperative, coordinate and negotiated activities, according to the information in model base and on blackboard.

The conflict resolution module is responsible for recognizing and resolving the conflicts among distributed heterogeneous cooperating agents. Negotiation, which is a very important issue in the domain of OOA, is used as a strategy for resolving conflicts. In ESA, there are three conflict types: knowledge conflict, constraint conflict, and perception conflict. Each conflict is resolved with a different negotiation strategy.

The control module is the interface to the problem solving layer. It is used to control and manage the activities of domain problem solving.

c. Problem solving layer

The problem solving layer is composed of a set of modules which relate to different domain problem solving tasks. Each module has its own knowledge-base and inference engine and can solve a special domain problem.

(4) Task Decomposition and Allocation

In STSS, each agent performs a class of performance. Real-time processing agent is responsible for the real-time tasks such as system's input and output, network calculation, and fault simulation, etc. User interface agent facilities instructor to interact with system. Fault diagnosing and fault restoration agents deal with complex and consuming time fault processing tasks.

After determining system's overall composition, next step is task decomposition and allocation among agents. There are two types of equipment in the simulative transformer substation. One is controller, and the other is protective relay. While controller can be further divided smaller units, such as generator, transformer bus, transmission line, etc. According to the hierarchy decomposition relationship, some of tasks can also be divided into several smaller subtasks.

For example, we decompose fault diagnosis task into three subtasks, and combine case-based reasoning with conventional problem solving paradigms. The first one is used for determining the blackout area based on dynamic variable with respect to network topology and breakers' status. The second one is used for determining fault equipment based on the fault rules of power equipment. And the last one is used for determining the fault components based the case with respect to protective relay. The diagnostic result of one level is the input of its next level. The diagnostic range is progressively reduced until the fault component is found. The relationships of these tasks are shown in Figure. 4.

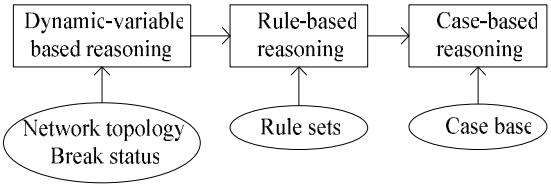

Fig.4: The relationships of fault diagnosis tasks.

(5) Multi-Agent Coordination and Cooperation

Coordinating the activities of multiple problem solvers is widely regarded as the central problem of OOA research. A number of approaches to coordination have been developed. They have ranged from very statically-defined models to models that can dynamically change with environments. In engineering application domain, choosing between coordination mechanisms for a particular application is a matter of system design. There is no universally best approach. Which method is chosen to achieve the balance between stability and flexibility is very important. Generally three mechanisms which are common in OOA are: organizational structuring, exchanging meta-level information, and multi-agent planning. we adopt a two-level-priority-based asynchronous communication approach which combines organizational structuring with exchanging meta-level information methods to realize the coordination among agents. For the interactions with respect to different functions, we statically define their priorities according to their

important extents. For example, diagnosis message which real-time processing agent receives from diagnosis agent is prior to setting message from user interface agent. While for the interactions which belong to a certain function and change with the environment, their priorities are calculated dynamically according to their urgent extents. For example, when multiple faults occur simultaneously, since one fault under different environment conditions has different urgent extents and it is difficult for fault simulation module to determine its order in the fault set, therefore, this dynamic process, which deals with a lot of complex knowledge about logical relationship between environment condition and fault, is processed by fault diagnosis module is more suitable than by fault simulation module [see Table 1]. We call the former the first level priority and the latter the second level priority.

Source agent	Goal agent	Message type
RT	UI	Switch status, load flow, conflict
	FD	Switch status, island
	FR	Switch status
UI	RT	Setting conflict, maintaining
	FD	Maintaining
	FR	Maintaining
FD	RT	diagnosis
	UI	diagnosis , maintaining
	FR	diagnosis
FR	RT	
	UI	Operation ticket
	FD	

Table.1: Messages exchanged among agents.

Agents in the cooperative problem solving system incoming messages in its communication handler to be executed. Generally, there tow disjoints types of messages, one is strictly used in communicative acts for initiating actions, and the other is used in response to former acts. Selecting which one to execute first is important for cooperative problem solving. In STSS, it is according to message's two level priorities.

In STSS, a message is composed of two divisions, head and body. For example, the format of a fault message is:

$$(\underbrace{id, pm, pn,}_{head}\ \underbrace{name, section, type}_{body})$$

Where id is the fault identifier; pm is the first level priority; pn is the second level priority; $name$ is the name of fault; $section$ is the fault section; $type$ is the fault type.

When there are two messages, such as A and B, in the communication handler of real-time processing agent waiting to be executed, the following three cases should be considered:

Case1: $id_A \neq id_B$

Case2: $id_A = id_B$ and $pm_A = pm_B$

Case3: $id_A = id_B$ and $pm_A \neq pm_B$

In case 1, because the identifiers of A and B are not equal, the function of message A is different from that of B and their first priorities must be unequal. The message that has bigger pm value will be first executed.

In case 2, because both id and pm values of message A and B are equal, the senders of message A and B are from the same agent. The message that has bigger pn value will be first executed.

In case 3, because message A and B have the same id value and different pm values, they come from different agents. After processing the message that has bigger pm value, the current environment, which may have been changed by the execution of this process, should be checked so as to determine if it can satisfy the requirement of another message which has been not executed. If environment satisfies the requirement, then execute the second message otherwise abandon it and inform its sender.

4. Conclusions

The main intention of this work is to offer a practical approach to OOA and knowledge engineering to support both with a complete tool environment and an example is introduced. Further research is planned in the following directions: Refinement of the method with knowledge engineering features; Improvement and extension of the expert supporting component of ESA; Integration of a facility that simulates the behaviour of objects to enhance the dynamic aspects of the system; Expansion of the ESA tool to a full programming environment, supporting OOA, OOD and both programming and debugging; Adaptation of the objects to standardized objects, as intended by groups like the OMG (Object Management Group), in order to cooperate with objects of different origin.

References

[1] P.L.George, M. A.Nicholas, and K P.George, Development of distributed problem solving system for dynamic environments. *IEEE Transactions on System and Cybernetics*, 18:400-414, 2005.

[2] R.J. Abbott, Program design by informal English descriptions. *Communications of the ACM*, 103:882-894, 2003.

[3] B. Alabiso, Transformation of data flow analysis models to object-oriented design. *Theory and Praxis*, 42:335-353, 1998.

[4] L. Gasser, An over view of object-oriented analysis. *Theory and Praxis*, 37:9-29, 2002.

[5] R.J. Nick, Controlling cooperative problem solving in multi-agent systems using joint intentions. *Artificial Intelligence*, 46:195-240, 2005.

[6] P. Desbiens, Design and operation of a virtual reality operator-training system. *IEEE Transce-nds on Power Systems*, 96:1585-1591, 1996.

[7] W. Mettrey, A comparative evaluation of expert system tools. *Computer*, 26:19-31, 2001.