УДК 681.3.06

# ПО НАПРАВЛЕНИЮ К УПРАВЛЯЕМОМУ СИНТЕЗУ ПРОГРАММ

**© 2011 г.** ***А. М. Абрамович***

***Ведущая Государственная Лаборатория по Разработке Программного Обеспечения, Уханьский Университет, Китай***

*В статье рассмотрен синтезатор программного обеспечения, который извлекает семантику из спецификации заказчика, переводит её в машинно-читаемую форму и создаёт новое программное обеспечение на основе готовых к использованию фрагментов кода.*

Ключевые слова: *управляемый синтез, представление знаний предметной области, целевая деятельность, конструктивные элементы, синтаксис менеджера.*

*A software synthesizer, which is able to extract a semantics from customer's specification, transforms it into machine-readable form, and creates a new software, basing on ready-to-use parts of code is presented in the article.*

Key words: *guided synthesis; domain knowledge representation; target activity; constructive elements; manager syntax.*

## 1. Introduction

*"AI researchers are interested in studying automatic programming for two reasons: First, it would be highly useful to have a powerful automatic programming systems that could receive casual and imprecise specifications for a desired target program and then correctly generate that program; second, automatic programming is widely believed to be a necessary component of any intelligent system and is therefore a topic for fundamental research in its own right."* [6]

There are at least three different software synthesis approaches:

1. Step-by-step improvement of programs (i. e. given specification has to be rewritten until its text takes the form of an executable program),

2. Synthesis by examples of pairs the input and target data or by examples of calculations,

From the example:

"1 gives 1, 2 gives 8, 3 gives 27 and so on" is it possible to derive a program for $y=x * x * x$, where $x$ is input data, and $y$ is output of the program, (the asterisk is "times")

3. Synthesis under the proof of the theorem that the required solution of a problem exists (deductive synthesis approach, "in which the derivation task is regarded as a problem of proving a mathematical theorem" [10]).

Deductive synthesis is described by the following scheme:

*task→ theorem → proof → program*

The deductive approach in its classical form requires for each subject area the availability of a complete list of assumptions that could be regarded as axioms of this domain. Their existence (include a priori given rules of inference) ensures the completeness of the models, allows posing and solving the range of problems related to the completeness, effectiveness and consistency of using models and procedures.

However, the recent Semantic Web's, Semantic Computing's, ERP's and others technologies don't provide a possibility of constructing axiomatic systems with a proper fullness. The inference, based on non-complete axiomatics, entails non-monotony processes of reception of results, an appearance of conflicts with statements received before as well as decreases the reliability of propositions resulting from a coherent process of logical inference. Thus it arose the problem of replacing the

formal system, with its procedures of deductive inference by other, equally powerful model, where the effects would be the main features of finding a solution to the ill-defined domains, which are described as open systems with updated knowledge about their structure and functioning [4].

There are attempts to strengthen the position of deductive synthesis using heuristics [18], restricted by scope of single strictly defined subject area (domain-specific deductive program synthesis [13]) and others.

The trend of limiting the program synthesis by scope of rigid-defined software environment, which is driven by high-level specification, presents a special interest. "Program Synthesis by Sketching", for example, suggested by Armando Solar Lezama, applies this idea. In Sketching, insight is communicated through a partial program (a sketch) that expresses the high-level structure of an implementation but leaves holes in place of the low-level details [5].

Following D. Pospelov, we represent knowledge not in the form of axioms, but as the detailed description of experience in the area of satisfaction of needs of separate subject domains (or needs of all society as a whole). Human experience evolves with a society and cannot be formalized once and for all. The offered software synthesis approach is based on knowledge representation system, described in [1], and can be carried out under the specification of the customer, which is written on semantically marked natural language.

Instead of proving the existence of solutions, we completely rely on the proven experience represented by domain expert and on the scenario of problems solving, proposed by him in his specifications.

Generally speaking, the proposed method is a kind of deductive analysis in the sense that the system's engine searches in the knowledge base a prototype of the task and adapts the found solution for the given private conditions.

We build domain knowledge data bases using formalism suggested in [1]. A customer specifies a task for the system synthesizer as a plan of his needs' satisfaction in detail or in a general form (in scope of his competence) by means of a simple syntax, which we name manager syntax, since it represents a customer's private method of resource management.

A system's synthesizer transforms the customer's specification into a specification in terms of the subject domain and finds a general solution using domain knowledge base, and applies it to the given situation. In the event that a suitable general solution is not found, a synthesizer looks for in the knowledge base constructive elements to meet the customer needs, and generates their composition as a new software solution.

The rest of paper is structured as follows: we give an overview of the Domain knowledge representation approach in Section 2; we introduce the manager's syntax in Section 3; we describe in brief a basis of Guided Synthesizer in Section 4; Section 5 contains a small example that clarify the manager syntax and a guided synthesis process.

## 2. Domain knowledge representation

According to [1] we consider t*arget knowledge as knowledge which the target system operates with the purpose of the given need*.

*By environmental knowledge* we mean knowledge about the environment that both motivates and governs the existence of target system. This knowledge about *an external environment contains cumulative knowledge of all external factors that influences on the target system's life cycle or/and knowledge about external processes (activities) that produces the environmental needs (optional)*.

Environmental need is knowledge about an environmental situation and environmental target activities (optional).

*By target activity we mean, generally speaking, a human activity aimed to the certain social need satisfaction*.

Target knowledge contains Human activity representation in theoretical (as a generic pattern) form and as its private implementations. Finalized target activity we denote here as Human experience. Strictly speaking, any target activity represents Human experience (mental or acquired in practice).

From the management's point of view target knowledge is knowledge about resources of target activity and configurations of constructive elements that defines an order of the target activity's implementation.

Figure 1 shows a Needs driven domain operating knowledge representation scheme.

Any social founded need (*need*) is associated with one of more ways of its satisfaction by means of a certain target activities. Therefore we unite all known ways of every separated need's satisfaction together and call such segment of domain knowledge by Need Satisfaction Domain.

*Need Satisfaction Domain unites knowledge that represents all known ways of certain need's satisfaction.* Need Satisfaction Domain is generated automatically as Need Language program [1; 21; 22; 23; 24] in response to the client's requirements and contains the following:

– ad of need,

– semantics of need,

– available resources or instructions for their obtaining,

– an operative plan of the need satisfaction.

As a rule, Need Satisfaction Domain integrates knowledge of different domains.

Every target activity belongs to a certain target environment and is represented by its resources, by configuration of constructive elements, by known situations, related to the target activity execution process.

Target activity is composed *by target activities' constructive elements and other resources.* We distinguish two kinds of target constructive elements, namely, *target subjects and target objects.*

*The target subjects are active participants of the target environment* (governments, enterprises, communities, families, persons, software agents and others).

*The target objects are passive participants of the target environment.* They are target environment's entities, which behavior is forced by target subjects.

Represented domain knowledge is considered as generic. Accordingly, its semantics and ontology is considered as generic too. System engine trough query-answering interface defines the mapping of semantics and ontology of the customer with generic ontology and semantics.

Note that, by domain ontology we mean a net of semantically marked domain concepts that map social needs, domain needs and the target activity situations.

**3. Manager syntax**

We believe that a customer, as well as all other people, is a manager of his life activity and related resources. Moreover, we believe that human's thinking (in the degree accessible to our understanding) is a management of intellectual resources. Therefore we consider a customer as a manager who manages both his own resources and resources provided to him.

Suggested system of automatic synthesis provides a manager with the possibility to specify a plan, which by his opinion is the best to meet his needs.

As shown in the Figure 2 the manager syntax allows specifying the following:

– Name of the need;

– Origin of the need (optional);

– Current state of the operational environment;

– The need satisfaction's available resources;

– Name of the solution's method (optional);

– Plan of a manager's needs satisfaction.

If a manager represents a current situation informally, its description, as a rule, includes the list mentioned above. A formal problem statement, usually, includes only a need's formulation and describes a current state of the operational environment.

We believe that a manager, interested in the

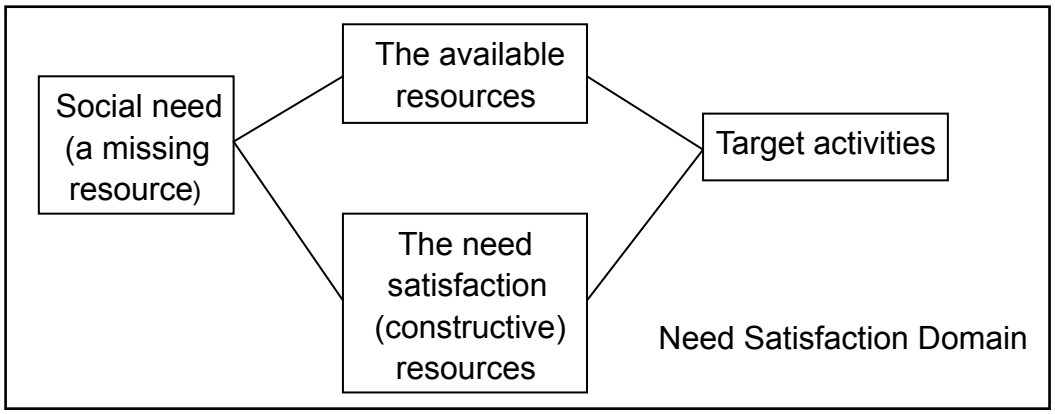



**Fig. 1.** Need Satisfaction Domain

result will describe his situation informally and use the following manager syntax in full:

**Manager** [Name]**: need:** *name_of_need*
**state:** *s1, s2, ... sn*
**causer:** *name_of_causer* |
*causer_description* |
*causer_environment*
*causer*
**resource:** *ar1, ar2,... ark*
**method:** *name_of_method*
**plan:** *[oper-tag]*
*[(]*
*[label:] name_of_sub-need* |
*[[,* | **ll***]*
*[label:] name_of_sub-need]*
*[)]* |
*expression* |
*[oper-tag]* |
*Plan*

Here *si* stands for an element of the operational environment's ontology, *arj* is an element of the available resources' ontology, *oper-tag* stands for **if**, **then**, **else**, **while**, **until**, **goto**, «**,**» (comma) means that the next sub-need must be satisfied after the previous, «**II**» (parallelism) means that sub-needs, shown in parentheses, can be performed in any order.

The *s1, s2, ... sn* are interpreted as symptoms of a current situation (i.e. list of subjects, objects, processes and their characteristics).

As shown in Figure 3, by causer of the situation we mean the following:

A causer (i.e. an event, a process, an object, a subject, a criminal, etc.) always acts in a customer's environment.

As a customer's environment may be the following:

– any animate or inanimate object,
– a professional or a private activity,
– a life activity as a whole.

A causer and a customer's environment are described by a special syntax.

We allocate hereafter with a bold the type terms of the syntax that named semantic tags. The rest components of statements a manager writes on his slang.

Therefore it is possible to say that the manager describes his statements in semantically marked natural language.

Note that a customer isn't obliged to know the manager syntax.

Analyzer's Query-Answering mode (see below) may capture all necessary information within the framework of interaction with a customer.

*3.1. The queries' templates*

To facilitate a customer interaction with a

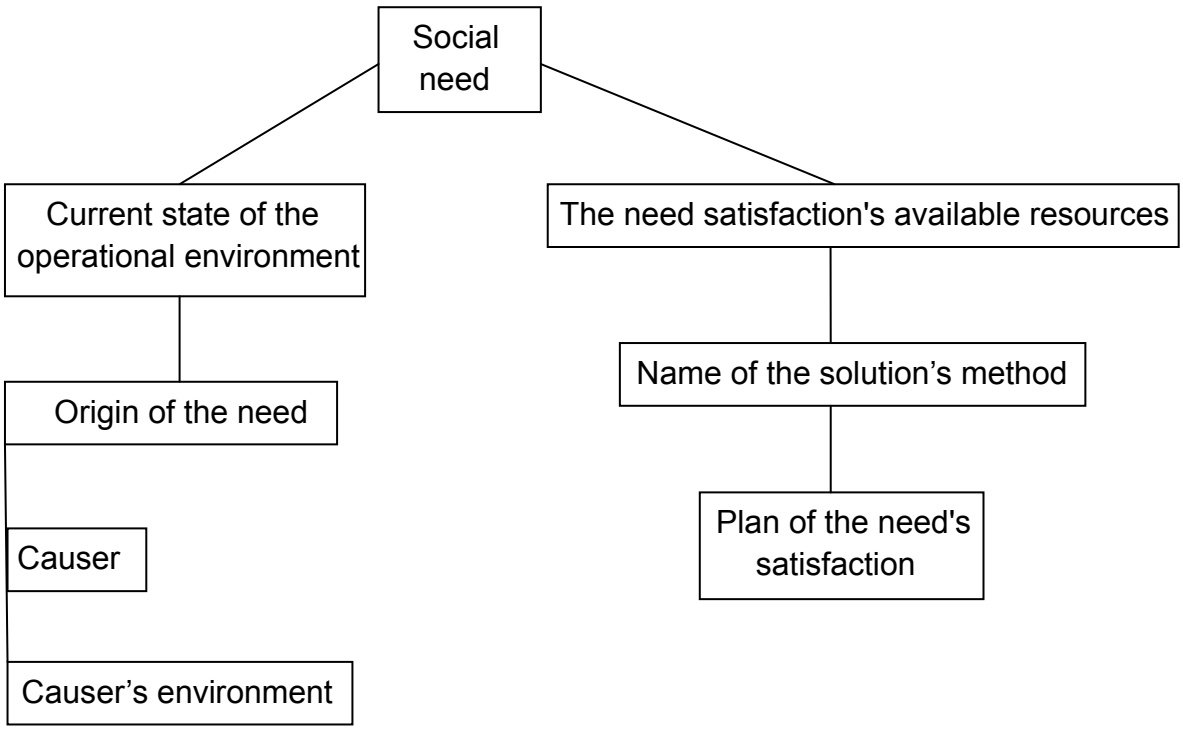



**Fig. 2.** Manager's semantic framework.

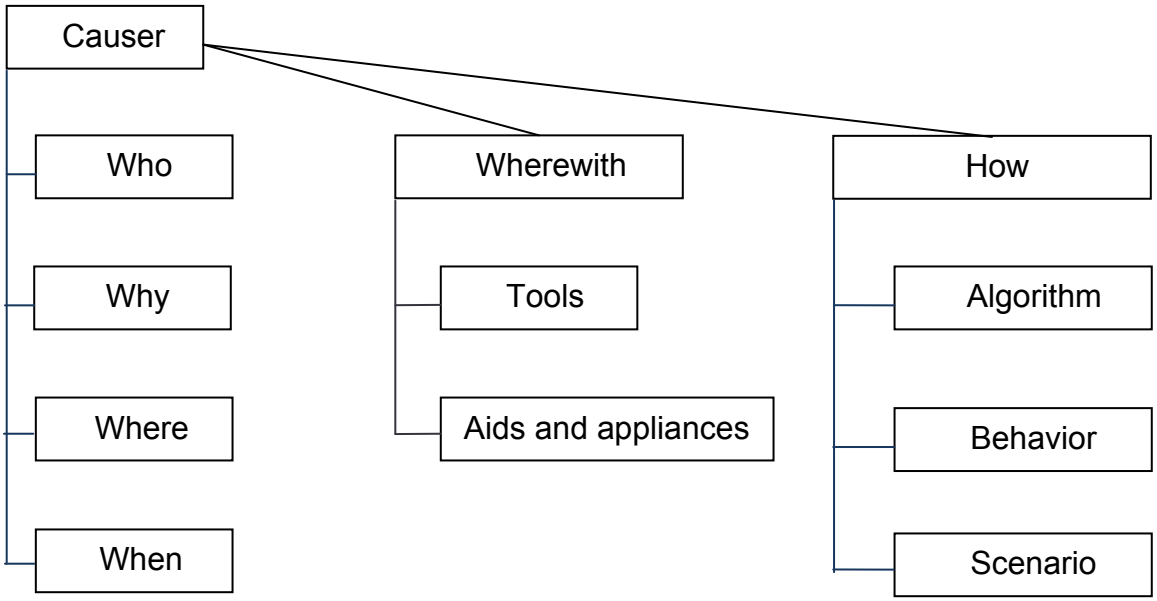



**Fig. 3.** Causer of the situation

system, the following templates of frequently asked questions are provided:

1. What is X?
2. How did you obtain X?
3. Why did you obtain X instead of Y?
4. Why did you obtain X?
5. How does X?
6. How do you apply Y?
7. What I must possess for X obtaining?
8. What target activity does satisfy my need N?
9. What resources are necessary for the target activity T implementation?
10. What constructive elements are included to the target activity T?
11. What is a contribution of the constructive element CE to the target activity T?
12. What is a repertoire list of constructive element?
13. What does constructive element CE?
14. and others

A manager may use these templates or formulate queries arbitrarily. A component OntoParser of Analyzer's Query-Answering mode (see bellow) translates a query's arbitrary text into internal domain knowledge base representation.

*3.2. The requirement's templates*

Manager may break in the calculation process using the following templates:

1. Give detailed answers for my questions
2. Display The Table of Conformity
3. Display the solution with detailed explanation
4. Display results of the sub-need SN's satisfaction Replace X by Y
5. Delete X
6. and others.

Requirements mentioned above are templates of frequently raised requirements. A manager may use these templates or formulate requirements arbitrarily. A component OntoParser of Analyzer's Query-Answering mode (see bellow) translates a requirement's arbitrary text into internal domain knowledge base representation.

**4. Needs driven Guided Synthesizer**

Figure 4 presents needs driven software synthesis process.

A manager guides the software synthesis process by means of his specification and by discussion with the Guided Synthesizer (synthesizer). The discussion clarifies the terminology and semantics of the specification. An ontology driven semantic analyzer (hereinafter Analyzer) is responsible for the discussion. It reads the specification step by step, makes a sentence's analysis that results a semantic expectation of the next sentence. If the expectation is confirmed, the Analyzer considers the next sentence. If not, it generates questions for the customer with the purpose to clarify the semantics. Analyzer takes upon itself a deduction

of the manager's specification by means of the Query-Answering mode in case the manager doesn't want using the manager syntax.

The Analyzer produces the "needs satisfactions resource" in form of known generic target activity or as a set of available constructive elements. After that Composer completes (alone or by discussing with a manager) preconditions and postconditions of operating constructive elements. This job results a sought-for Executable target activity.

Analyzer provides all basic abilities of an intellectual interface [2]. It not only guides the software synthesis, but also provides communication with the manager, proves and explains decisions, and trains users. Herewith it operates with knowledge from the Experience base. Therefore knowledge browsing is an important role of Analyzer.

Using the factual mapping relation [1], we describe the Analyzer by the following way:

*Analyzer ├ Knowledge Browsing, Communication, Software Synthesis, Explanation, Training*

Note that Analyzer acts on the basis of the understanding process.

We define here an understanding process as *a proved mapping of the manager statements' semantics into generic semantics of Experience base's statements.*

We mean by a manager's statement his specification, query, message or requirement.

*4.1. Communication with a manager*

Ontology driven semantic analyzer guides a communication with a manager by means of understanding of his professional slang. For this purpose it compares the semantics and ontology of the customer with generic ontology and semantics. This mapping is one of the main actions of Analyzer's Query-Answering mode. Query-Answering mode produces Correspondence Table of the manager's terms with generic ontology. For this purpose Analyzer generates questions by means of Need Language semantic framework [1; 21; 22; 23; 24].

For example, in case of unknown **need** it defines semantic coordinates [1] of the need by means of questions like the following:

– What is your aim?

– What is the motivation for your **need**?

Answers for such questions detect **need's** semantics.

In case of a manager's specification is semantically unclear, Analyzer finds conformity between separate units of the manager's specification and possible Need Satisfaction Domain's elements by means of questions like the following:

– Have you resource, named X (Def: X is …)?

– Has the resource X the attribute, named Y (Def: Y is…)?

And so on.

Analyser clarifies the X and Y semantics by means of declarative descriptions [1].

Based on Correspondence Table, the Analyzer "understands" the manager's requirements and messages.

Using Correspondence Table the Analyzer translates an initial specification into specification in generic ontology terms. It debates a new specification with a manager by means of Query-Answering mode. Final specification contains description of executable target activity and all operating constructive elements.

*4.2. Knowledge browsing*

Semantic marking of a manager's specification grounds a semantic search in Experience base. It is possible since both an experience's description and the manager syntax uses the same semantic tags [21]. Analyzer translates an initial specification into specification in generic ontology terms (applying the Correspondence Table) and uses a **need** as well as **plan** units as key words for discovering a target activity that meets the **need's** semantics and which constructive elements' semantics suits the **plan's** units.

Analyzer uses content of manager's specification as key words for the confirmation of found target activity as a sought-for activity as well as for the discovering of the necessary constructive elements.

In case of absent a target activity that suits the manager's specification, Analyzer sequentially finds in domain knowledge base constructive elements that suit the **plan's** units.

Generally speaking, we consider a manager's specification as a semantically marked requirement for search in the experience base. Herewith Analyzer processes the specification using Query-Answering mode for the retrieving

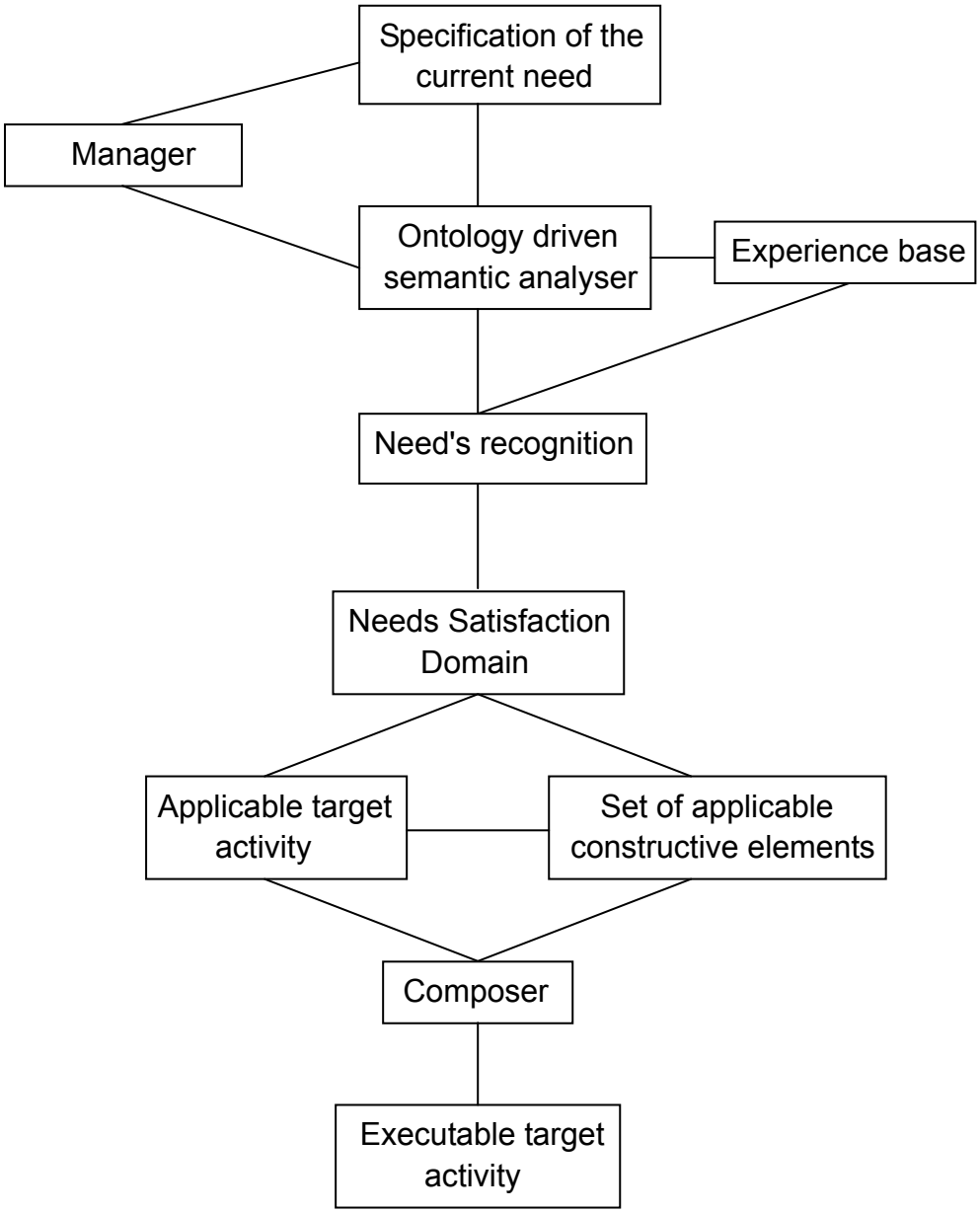



**Fig. 4.** Needs driven software synthesis process

the semantics of unknown terms.

*4.3. Synthesizer*

Following D. Pospelov [4], we consider the synthesizer (in his terminology – Solver) as a functional unit of the intellectual interface.

Extending the philosophical meaning of the understanding concept, we consider it as the solution concept. This implies that we identify a complete understanding with a complete solution.

Accordingly, understanding is the important function of Analyzer.

Analyzer transforms the manager's specification into the Need Satisfaction Domain specification by means of knowledge browsing and the communication process. If Analyzer discovers a generic target activity, which semantics and configuration satisfies the manager's specification, then it checks and fulfills all necessary preconditions and postconditions and declares the software synthesis process ended.

If not, Analyzer generates a new target activity that satisfies the manager's specification. For this purpose it searches target activities and constructive elements (both local and general) that separately satisfy the **plan's** units and composes sought-for software.

In case of a target manager's Need

Satisfaction Domain contains knowledge of software platform, Analyzer composes ready to use fragments of code that correspond to given domain entities.

According to domain knowledge building recommendation [1] the best way to build Experience base effectively is to represent all its components by means of unified software platform. In this case all units of a manager's specification will be translated by Analyzer to the code fragments automatically.

4.4. Grounding and explanation

Grounding and explanation is a mandatory mode of an intellectual interface. This mode provides a manager with detailed answers for the following types of queries: "What X is?", "How did you obtain X?", "Why did you obtain X instead of Y?", "Why did you obtain X?", "How does X?", "How do you apply Y?", "What do I need to get X?", "What target activity does satisfy my need N?" and other. Grounding and explanation's mode also satisfies such the manager's requirements like "Display the Correspondence Table", "Display the solution with detailed explanation", "Teach me how to satisfy the need " and others.

The answer for question may be detailed in accordance to the manager's demand "Give detailed answers for my questions". For example, as the answer for question "What is a triangle?" may be a definition "A triangle is a part of the plane formed by the intersection of three lines" or may mirror software representation of a triangle. For example, manager will receive definitions that mirror methods of programming's representations of a triangle, which are accepted in the experience base under the Need Language's notation:

**Def:** Triangle V1V2V3
General:
(vertex(3), side(3), angle(3))
**Vis.:** General introduces the general Triangle's model,
**Vis.:** vertex is a pointer to array that contain 3 triangle's vertexes
**Vis.:** side is a pointer to array that contain 3 triangle's sides
**Vis.:** angle is a pointer to array that contain 3 triangle's angles (x1,y1), (x2,y2), (x3,y3):
**Vis.:** where (x1,y1), (x2,y2), (x3,y3) describes a Triangle's model,
**Vis.: Vi** names the triangle vertex **i**
**Vis.: (xi,yi)** means coordinates of triangle's vertex **i**
and so on.

For example, in case of the need is a calculation of the spatially-rod constructions (that consist of pyramids and triangles) by the finite elements method, a manager receives a definition of triangle from the calculation of stress-strain state's point of view (i. e. (x1, y1), (x2, y2), (x3, y3) model of triangle).

Knowledge representation approach that is recommended in [1] facilitates answers for above mentioned queries and requirements due to the deep specification of data. The deep specification of data contains declarative descriptions and detailed information about environmental needs and target activities that allows deriving the target causal-effect network that covers the target activity's environment.

Interpreting the structure of represented knowledge, Analyzer generates a target causal-effect net in the following way: the external environment need causes the target activity; the need of the target activity causes the need of applying the set of constructive elements and so on. Using the causal-effect net, Analyzer acts as a navigator with the purpose of the providing both groundings and explanations.

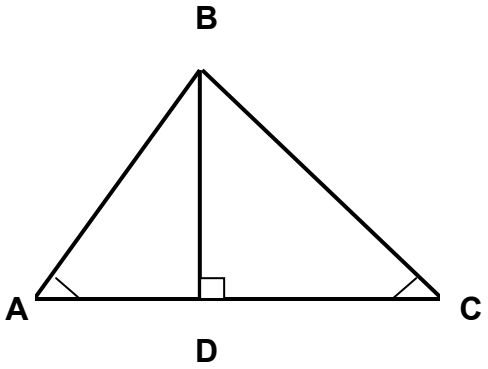
The cause and effect network provides also understanding process. It provides a building and verification of hypothetic interpretations of a manager's specification. Using the causal-effect net, Analyzer builds expectation (hypothesize) of the manager specification's semantics. This semantics will be proved or rejected further in the course of both semantic search in Experience base and Query-Answering process.

## 5. Example of manager's specification. Calculate the perimeter of the triangle

Note that for clarity, we use here the usual denotations of the triangle and all its components. In practice, the manager will use his own terminology.

It's given $\triangle ABC$ such that $\angle A = 58°$, $\angle C = \triangle 55°$, $BD \perp AC$, $DC = 4cm$.

Find perimeter of a triangle.




Specification 1 (the manager writes detailed specification):

**Manager:**

**need**: Perimeter △ABC

**state:** ∠A=58°, ∠C=55°, D∈AC, BD⊥AC, DC=4cm

**plan**: BC, AB, AC, Perimeter △ABC

**need:** length(BC):

**state:** △BCD, DC=4cm, ∠D=90, ∠C=55°

**need:** length(AB)**,** length(AD)

**state:** △ABD, A=58**,** D=90, ∠BD

**need:** Perimeter: △ABC

**state**: AB, BC, AC=AD+DC

As a response to above specification Analyzer searches in the domain knowledge base an appropriate target activity (using **need** and **plan**). Since the Experience base doesn't contain a target activity that suits **need** and **plan**, Analyzer consequentially satisfies the manager's sub-needs listed in **plan**.

For the purpose of length (BC) calculation (i. e. length of BC) Analyzer discovers that △BDC is a right triangle, builds (using a corresponding generic object's model) its model, and applies one of methods (local target activity) attached to this model that calculate a hypotenuse (BC) as a quotient from division of a cathetus (DC) of a triangle on cosine a corner adjoining to it (∠C).

The same actions lead to the calculation of AB.

For the purpose of Perimeter calculation Analyzer builds a model of △ABC, finds a local activity that calculates its perimeter as sum of sides lengths.

Specification 2 (the manager writes non-detailed specification):

Manager:

**need**: Perimeter △ABC

**state**: ∠A=58°, ∠C=55°, D∈AC, BD⊥AC, DC=4cm

Processing this specification, the Analyzer builds the models of △ABC, △ABD and △BDC, and finds above mentioned local activities independently.

**6. Conclusion**

Generally speaking, an automated software synthesis task resolves itself into an automated thinking's task.

Since the nature of thinking is still unknown, all that we can in these circumstances – is a detailed representation of the human thinking's results that are fulfilled in practice (human experience), as well as an advanced semantic search mechanism to capture knowledge for the problems solving. Herewith we consider knowledge of human experience as a resource of the thinking process.

Suggested synthesis approach starts from understanding a customer as a manager of available knowledge resources, a representation of human experience in form of Need Satisfaction Domains and a query-answering driven search mechanism.

*The guided synthesis is able to produce both software (using previously represented fragments of code) and scenarios of human activities for the performance by a manager independently or by means of his staff.*

*The guided synthesis releases a manager (i. e. a customer) from routine operations. Using his specification (detailed or not) executed in the professional slang, the system generates internal specification using domain ontology, and composes a detailed description of the target activity that ready for execution.*

**References**

*1. Abramovich, A.* "Domain knowledge representation". Wuhan University, 2009.

2. *Pospelov, D.* Models of Human Communication: Dialogue with Computer. // International Journal of General Systems. – 1986. – Vol. 12, – №4. – P. 333–338.

3. *Поспелов Д. А*. Электронная вычислительная техника. Сборник статей. Вып. 3. – М.: Радио и связь, 1989. – С. 4–20.

4. *Поспелов Д. А*. Десять «горячих точек» в исследованиях по искусственному интеллекту. // Интеллектуальные системы (МГУ). – 1996. – Т. 1, вып. 1–4. – С. 47–56.

5. *Lezama, Armando Solar.* Program Synthesis by Sketching, Technical Report No. UCB/EECS-2008-177, http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-177.html.

6. *Biermann, A.* Automatic Programming. In Encyclopedia of Artificial Intelligence. 2nd edition. – New York: John Wiley&Sons, 1992.

7. *Liu, H., Lieberman, H.* Metafor: Visualizing Stories as Code, http://larifari.org/writing/IUI2005-Metafor.pdf.

8. *Liu, H., Lieberman, H.* Toward a Programmatic Semantics of Natural Language. Proceedings of the 20th IEEE Symposium on Visual Languages and Human-Centric Computing. IEEE Computer Society Press, 2004. http://larifari.org/writing/CHI2005-NLInterfaces.pdf.

9. *Fisher, B., Schumann, J.* Automated Synthesis of Statistical Data Analysis Programs, Proc. SDP'02: Workshop Science Data Processing, Greenbelt, MD, Feb. 26–28, 2002.

10. *Fischer, B., Pressburger, T., Rosu, G., Schumann, J.* The AutoBayes Program Synthesis System – System Description, Proc. CALCULEMUS 2001: 9th Symp. on the Integration of Symbolic Computation and Mechanized Reasoning, Siena, Italy, Jun. 21–22, 2001.

11. *Fischer, B., Schumann, J.* AutoBayes: A System for the Synthesis of Data Analysis Programs, Proc. NIPS 2000: Workshop on Software Support for Bayesian Analysis Systems, Breckenridge, CO, Dec. 1, 2000.

12. *Buntine, W., Fischer, B., Pressburger, T.* Towards Automated Synthesis of Data Mining Programs, Proc. KDD'99: 5th ACM Int'l Conf. on Knowledge Discovery and Data Mining, San Diego, CA, Aug. 15–18, 1999.

13. *Fischer, B., Pressburger, T., Rosu, G., Schumann, J.* The AutoBayes Program Synthesis System-System Description, RIACS / zCode IC, NASA Ames Research Center, M/S 269-2.

14. *Whittle, J.* and others. Amphion/NAV: Deductive Synthesis of State Estimation Software. ASE Group: NASA, http://ti.arc.nasa.gov/m/tech/rse/publications/papers/ASE01/final.pdf.

15. *Manna, Z., Waldinger, R.* Fundamentals of Deductive Program Transactions on Software Engineering. – 1992. – Vol. 18(8). – P. 674–704.

16. *Smith, Douglas R.* A Problem Reduction Approach to Program Synthesis. http://dli.iiit.ac.in/ijcai/IJCAI-83-VOL-1/PDF/005.pdf.

17. *Демьянков В. З.* Понимание как интерпретирующая деятельность. // Вопросы языкознания. – 1983. – №6. – С. 58–67.

18. *Lezama, Armando Solar.* Program Synthesis By Sketching, EECS Department, University of California, Berkeley, Technical Report No. UCB/EECS-2008-177, December 19, 2008. http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-177.pdf.

19. *Korukhova, Yulia S.* Automation of Program Synthesis from Logic-Based Specifications in the Deductive Tableau, Lomonosov Moscow State University.

20. *Abramovich, A.* Need Language knowledge representation platform, Wuhan University, 2009.

21. *Abramovich, A.* Need Language, COLLIN, 2010.

22. *Abramovich, A.* Towards a Global Platform of Human Experience, SDPS, 2010.

23. *Abramovich, A.* Towards Linked Needs, ICSC, 2010.

*Поступила в редакцию* *18 января 2011 г.*

**Александр Марианович Абрамович** – доктор математических наук, исследователь Ведущей государственной лаборатории по разработке программного обеспечения Уханьского университета (Китай). Автор более 50 научных публикаций. Руководитель международного исследовательского проекта «Связанные потребности».

**Alexander Marianovich Abramovich** – M.Sc. in mathematics, visitor researcher (State Key Laboratory of Software Engineering, University of Wuhan, China). Author of more than 50 scientific works. Leader of International research project «Linked Needs».

38/13 Hazionut st., 35312, Haifa, Israel
Тел.: 972-502-1944-36, e-mail: webdao@gmail.com

---

**I ВСЕРОССИЙСКИЙ СИМПОЗИУМ ПО РЕГИОНАЛЬНОЙ ЭКОНОМИКЕ**
**(Екатеринбург, 28 июня – 1 июля 2011 г.)**

Секция экономики Отделения общественных наук РАН, Институт экономики УрО РАН, Российский фонд фундаментальных исследований, Российский гуманитарный научный фонд, ОАО «Трубная металлургическая компания» при информационной поддержке Журнала экономической теории, журнала «Экономика региона» и издательского дома «Финансы и кредит» извещают о проведении I Всероссийского симпозиума по региональной экономике, посвященного 40-летию Института экономики УрО РАН.

Председатель Организационного комитета Симпозиума – академик А. И. Татаркин. Сопредседатели – академик П. А. Минакир, академик В. В. Кулешов. Заместитель председателя д.э.н. Ю. Г. Лаврикова. Ученый секретарь – к.э.н. М. В. Власов.

**Работа Симпозиума будет проходить по следующим научным направлениям**:

1. Направления и проблемы развития современной теории и методологии региональной экономики.
2. Институты регионального инновационного развития
3. Институты саморазвития территорий разного уровня.
4. Инструментарий и методы прогнозирования регионального развития.
5. Современная государственная региональная политика.

**Регистрация участников** осуществляется ТОЛЬКО на сайте Института экономики УрО РАН: www.uiec.ru, заявки и тезисы докладов, присланные любым другим способом, Оргкомитетом рассматриваться не будут.

Адрес Оргкомитета симпозиума:
620014, Екатеринбург, ул. Московская, 29, Институт экономики УрО РАН, каб. 513.
E-mail: simpozium2011@mail.ru
Адрес сайта для регистрации участников, отправки заявок и тезисов докладов: www.uiec.ru