Hirashima,T. et al. (Eds.) (2010). Workshop Proceedings of the 18th International Conference on Computers in Education. Putrajaya, Malaysia: Asia-Pacific Society for Computers in Education.

Question and Problem Generation – State of The Art

Nguyen-Thinh LE[a], Tomoko KOJIRI[b]
[a]*Clausthal University of Technology, Germany, *
[b]*Nagoya University, Japan*
nguyen-thinh.le@tu-clausthal.de, kojiri@nagoya-u.jp

Abstract: Currently, there are two communities which focus on question and problem generation. The first one has been starting from 2006 and runs workshops at the ICCE conference, whereas the second one has been established since 2008 and has been hosted at AIED2009 and ITS2010. Through these two communities, numerous approaches to generating questions and problems have been devised. The goals of this paper are to summarize the state of the art in this research area.
Keywords: question/problem generation, applications

Introduction

Currently, researchers from multiple disciplines have been showing their common interest in question and problem generation (QPG). The natural language processing community is interested in the methods of how to extract questions from a natural language text. The intelligent tutoring systems community deals with generating both questions and problems which are adaptive to the knowledge state of individual students. Researchers from the education community investigate mainly the question of how and whether questions may improve the learning process of students. In this paper, we summarize the state of the art of QPG. Our discussion will focus on the applicability of QPG in educational systems. The paper distinguishes between question and problem generation. We will answer the following questions: 1) Why is QPG useful? 2) Which methodologies can be applied to generate questions/problems and how automatic question/problem generation can be deployed in educational systems?

1. What is Question and Problem Generation?

Question generation (QG) has been defined in [1] as a task of generating questions from some forms of input, e.g., from a deep semantic representation or a raw text. Most of current works in question generation concentrate on text-to-question and tutorial dialogue. Dale and White [4] distinguished between question generation from paragraphs and question generation from a sentence. Generating questions from paragraph is more challenging than from a sentence, because a paragraph contains causal relations between sentences (e.g., pronouns), which need to be elicited and annotated explicitly.

Similarly, problem generation is achieved by composing necessary information from a structured or unstructured database of domain specific knowledge and transforming the composition into a well-formed natural language text. The main topics in the most literatures of the problem generation are how to model domain knowledge and how to automatically generate problem components, such as given conditions, goal, and solution to the goal, based on the domain knowledge.

2. Question Generation

2.1 Question Classification

Before questions should be generated either from a text or from a structured database, the type of possible questions should be determined. Thus, question taxonomy is required to build a model for question extraction. An additional use of question taxonomy is for evaluating generated questions [5]. The most coarse question classification has been proposed by Kalady and colleagues [6]. The authors distinguished between factoid and definitional questions. Factoid questions are usually generated from elementary sentences of an input document and have short fact- based answers. On the contrary, definitional questions have a descriptive answer that exists within several sentences of an input document and can only be answered after the whole document has been read (or processed). An example of a definitional question is "*What is a volcano*?" A possible answer for this question may require more than one sentence. A more fine-grained question classification, which is widely used in existing automatic question generation systems, includes the following question types: *how, who, what, which, when, where and why* [7, 8]. In addition, *Yes-No* questions [6, 9] and quantitative questions [10] such as *how much/many, how old, what percentage of* have also been exploited for extracting questions. Researchers grouped question types *who, what, when, where* into the class of shallow questions, and *why, how, what-if* are considered deep ones [1]. Several complex question taxonomies have been developed in the area of education (e.g., in medical examination, engineering, and trial questioning) such as Socratic questions, the Bloom's taxonomy, and the Lehnert's and Graesser's taxonomy [5]. Rarely, these question taxonomies can be found in question generation systems. Boyer and colleagues [11] explained that a general question taxonomy does not meet the requirements of all application areas and of specific tutors.

2.2 Approaches to Automatic Question Generation

The problem of question generation can be illustrated using the following sample sentence: *"The 18th ICCE will be held between November 29 and December 3, 2010 at Putrajaya, Malaysia."* Possible questions are: *1) when will the 18th ICCE be held? and 2) where will the 18th ICCE be held?*
According to Becker and colleagues [12], the process of question generation has to involve in the following issues: 1) Target concept identification: which topics in the input sentence are important? 2) Question type determination: which question types are relevant? 3) Question formation: how questions can be constructed grammatical correctly? The first and second issues are usually solved by most QG systems in the similar manner using the following different techniques of natural language processing (NLP):
Parsing: The input sentence is analyzed in functional constituents (e.g., the noun phrase, verb phrase, adverb, preposition) according to the grammar of the language being used.
Simplifying sentence: Complex sentences can be simplified using text compression techniques [13]. Given a complex sentence, text compression techniques produce a single shortened version that conveys the main information in the input. Hailman and Smith [14] argued that text compression is not sufficient to generate high quality questions, because a complex sentence often conveys multiple pieces of information. As an alternative, the authors suggested a method to extract appositives, subordinate clauses, and other constructions from complex sentences using *semantic entailment* and *presupposition*.
Anaphor resolution: This technique replaces pronouns in a sentence by a concrete noun [6].
Semantic role labeling: The goal is to annotate each constituent of the input sentence with a semantic role in relation to the verb (also called the predicate) in the sentence. Semantic roles

can be classified into e.g., agent, patient, instruments and adjuncts (locative, manner, temporal). For instance, the sample sentence can be annotated with semantic roles using the Illinois Labeler [15], where A1, VERB, AM-MOD, AM-TMP, AM-LOC represent semantic roles: *[(thing held **A1)** The 18th ICCE] [(General modifier **AM-MOD**) will] be [(**Verb**: hold) held] [(temporal: **AM-TMP**) between November 29 and December 3, 2010] at [(location: **AM-LOC**) Putrajaya, Malaysia]*.

Named entity recognizing: The parsed constituents of the input sentence are assigned to unique entities (e.g., PER (person), LOC (location) or ORG (organization)). For instance, the sample sentence above will be tagged as followed using the Illinois named entity recognizer [16]: *The 18th [**ORG** ICCE] will be held between November 29 and December 3, 2010 at [**LOC** Putrajaya], [**LOC** Malaysia]*.

The results of the semantic role labeler and the named entity recognizer mainly contribute to solve the second issue, namely determining appropriate question types for the identified target concepts. For instance, if the entity in a proper noun phrase, then "*who/whom*" is suggested as appropriate question types for entity of type PERSON or ORGANIZATION, and "*where/which*" for LOCATION; if the entity is of type NO ENTITY, then the question type "*what*" is suggested [6, 17]. Besides the methods of determining question types based on the named entity recognizer and the semantic role labeler, other types of question can be identified using information of the parse tree, e.g., a prepositional phrase is a candidate for question [6]: *The dog is asleep on his bed => On what is the dog asleep?* And an adverb phrase can also be the target for a "How"-question: *The meeting went well => How did the meeting go?*

Where most QG systems share common techniques on the first and second step of the process of question generation, their main difference can be identified when handling the third issue, namely constructing questions in grammatically correct natural language expression. Many QG systems applied the transformation-based approach to generate well-formulated questions [6, 8, 10, 18, 19]. In principle, transformation-based question generation systems work through several steps: 1) delete the identified target concept, 2) a determined question key word is placed on the first position of the question, 3) convert the verb into a grammatically correct form considering auxiliary and model verbs. For example, the QG system of Varga and Le [19] uses a set of transformation rules for question formation. For subject-verb-object clauses whose subject has been identified as a target concept, a "*Which Verb Object*" template is selected and matched against the clause. "*Which*" replaces the target concept in the selected clause. For key concepts that are in the object position of a subject-verb-object, the verb phrase is adjusted (using auxiliary verb).

Table 1: Evaluation results of existing question generation systems

System	Question type	Evaluation Results
[6]	Yes-No, Who, Whom, Which, Where, What, How	Recall=0.68; Precision=0.46
[8]	Yes-No, Who, Which, Where, What, When, How, Why	Recall=0.32;Precision=0.49
[19]	Who, Whose, Whom, Which, What, When, Where, Why, How many	Relevance[1]=2.45(2.85); Syntactic Correctness & Fluency=2.85(3.1)
[17]	Who, When, What, Where, why, How	Low acceptance, no data avail.
[10]	Yes-No, Who, When, Which, What, Why, How many/much	Satisfactory results, no data avail.
[21]	Question templates for informational text	79.9% plausible questions[2]
[23]	Question templates for narrative text	71.3 % plausible questions

The second approach, which is also widely employed in QG systems is template-based [20, 21, 22]. The template-based approach relies on the idea that a question template can capture a class of questions, which are context specific. Chen and colleagues [21] developed the

[1] The evaluation criteria *Relevance* and *Syntactic correctness and fluency* are rated by from 1 to 4, with 1 being the best score. Values outside and inside in the brackets indicate ratings of the 1st and 2nd human.

[2] The evaluation results are calculated as the average of the plausibility percentage of three different question types: 86.7% (condition), 65.9% (temporal information), 87% (modality).

templates "*what would happen if* <X>" for conditional text, "*when would <X>*" and "*what happens <temporal-expression>*" for temporal context, and "*why <auxiliary-verb> <X>*" for linguistic modality, "*where <X>*" maps to semantic roles annotated by a semantic role labeler. These templates are specific for informational text. For another type of text, e.g., narrative text, Mostow and Chen [23] developed another set of question templates. Another technique to define question templates is using regular expressions [20]. Sneiders [22] developed question templates whose answers can be queried from a structured database. For example, the template "*When does <performer> perform in <place>?*" has two entity slots, which represent the relationship *performer-perform-place* in the conceptual model of the database. Thus, this question template can only be used for this specific entity relationship. For other relationship types, new templates need to be defined. Hence, the template-based approach is mostly suitable for applications with a special purpose. A disadvantage of this approach is that developing high-quality templates requires a lot of human involvement. Table 1 summarizes the evaluation results of different existing question generation systems. We notice that the template-based systems [21, 23] achieve considerable results, whereas the transformation-based ones [6, 10, 19] need more improvement.

2.1 Direction for Question Generation in Educational Systems

From the technical point of view, automatic question generation can be achieved using a variety of natural language processing techniques which have gained wide acceptance. Currently, high quality shallow questions can be generated from sentences [1]. Deep questions, which capture causal structures, can also be modeled using current natural language processing techniques, if causal relations within the input text can be annotated adequately. However, successful deployment of question generation in educational systems is rarely found in literature. Researchers rather focus on the techniques of automatic question generation than the strategies of deploying question generation into educational systems. A similar picture has been identified by Mostow and Chen [23] especially in the settings of training reading comprehension: most existing works in this field rather randomly choose sentences in a text to generate questions, than pose questions in an educational strategic manner.

Research on question generation with focus on educational systems needs to develop further. Several research questions can be raised: If the intent of the questions is to facilitate learning, which question taxonomy should be deployed? Given a student model in an intelligent tutoring system, which question type is appropriate to pose the next questions to the student? Apart from generating questions from an input text, questions may also be generated from other source of data (e.g., Wikipedia) to remind the student to additional information if necessary. The question is when and how additional data should be deployed? Another area of deploying question generation in educational systems may be using model questions to help students improve the skill of creating questions, e.g., in the legal context.

3 Problem Generation

3.1 Classification of Problem Generation

Problem consists of problem sentence, which describes given conditions in natural language, and its solution sentences, which represents the goal and its solution. In problem-solving process, problem sentences change their structures variously to the goal structure, which characterizes the different aspects of the problem. Problem structures in problem-solving process are modeled as Figure 1, which is a revised version of problem-solving process

proposed in [25] [3]. Surface structure describes given conditions of the problem, such as objects, attributes, and relations between them. Surface structure is transformed into the formulated structure and applied domain knowledge to derive the goal structure. Domain knowledge represents operations to change the objects and attributes. The knowledge includes conceptual knowledge of objects and attributes and numerical knowledge among them. Types of necessary knowledge are different according to the domain. The change of formulated structure to the goal structure with the domain knowledge corresponds to the solution structure. Natural language descriptions of the solution structure become solution sentences.

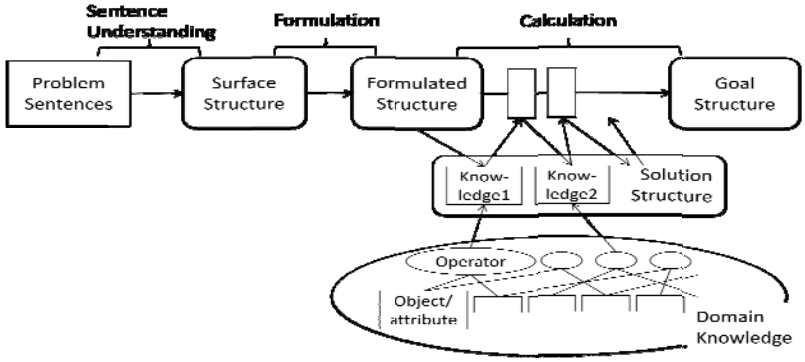

Figure 1: Model of Problem-Solving Process (Revised version of [25])

Problem generation is to generate particular structure/s of problem by given structure/s. It is classified as the following two categories: 1) To derive solution structure/solution sentences from problem sentence/surface structure; 2) To generate problem sentence/surface structure that satisfies given solution structure/domain knowledge. In the first category, the challenging issue is to model domain knowledge and method for finding the solution structure from given formulated structure automatically. Determination of validate surface structure that satisfies certain solution structure or domain knowledge is the main focus in the second category.

The difference of problem generation approach of the system depends on the types of problems that they generate. For the problems whose goals are to find one of the answers that satisfy given constraints, solution structures are not so important. In addition, they may have plural goals and to prepare all solution structures is not a realistic task. For this type of problems, second problem generation category, which focuses on preparing meaningful surface structure, is often applied. For problems that have one or only a few answers, it is important to select appropriate domain knowledge to derive the goal structure. For such problems, the first problem generation category is often observed. Details of these approaches are shown in Section 3.2 and Section 3.3, respectively.

3.2 Approaches to Derive Solution Structure as Problem Generation

Solution Structure is derived by applying domain knowledge to the formulated structure. The varieties of problems that can be generated by the system are determined by the size of its domain knowledge. Many systems are developed for different domains (Table 2). Most systems focus on the domain whose problem-solution process is clearly described. In these domains, goal structure can be generated by changing given objects and attributes to the required one based on the domain knowledge. Therefore, the domain knowledge is represented as operations/relations among all possible objects, attributes, and their values.

[3] Hirashima and colleagues [34] defined only numerical constraint as domain knowledge, since they focus on physics. We revised their figure so as to apply to the general problem-solving process.

Table 2: Targets of existing problem generation systems for deriving solution structure

System	Target domain
[26]	Mechanics
[27]	Chemistry
[28, 29]	Arithmetic

Hirashima and colleagues [26] proposed phenomenon structure as domain knowledge in mechanics. Phenomenon structure describes phenomenon as operational relations and attributes that relate to the phenomenon. For example, the phenomenon "*block falling by the gravity"* is represented as operational relation "the law of gravitation *F=m*g*" and attributes "mass of block *block(mass(m))*", "acceleration of gravity *block(gravity-acceleration(g))*", and "gravitation for block *block(gravitation(F))*". If the goal of the problem is to derive the gravitation of the block and mass of block is given as *m1* and acceleration of gravity is given by *g*, the system is able to derive the gravitation *m1*g* by applying this operational relation. In this approach, the phenomenon structure takes the role of the rule for changing the attributes of objects. Ishima and colleagues [27] also applied this concept into the chemistry. It prepares knowledge on material, knowledge on phenomenon and numerical relation knowledge related to them. Since it models the chemical world from various aspects, it is able to derive the solution structure of various types of problems, e.g. problems that ask the value of attribute.

Matsuura and colleagues [28] modeled attributes of objects and their numerical relations statically as one directed graph. It consists of two types of nodes; *index of variable* that represents attributes of the objects and *relation of variable* that indicates numerical relations between the attributes. Links connect these two types of nodes. When the surface structure is provided, given values are assigned to corresponding *index of variable* initial state and *index of variable* of required value is set as a goal. To derive the solution structure is to find the path from the initial state to the goal by replacing unknown values in *index of variables* using *relations of variable*. Kojiri and colleagues [29] introduced solution network to represent the relations of problem and sub-problem. The solution network connects templates of problems based on the inclusive relations. The solution structure is determined by following the solution network, from the problem that has required attribute as its goal until its given attributes are all derived.

3.3 Approaches to Derive Surface Structure as Problem Generation

Surface structure defines situations of problem in which specific domain knowledge can be applied. Plural surface structures exist for the same set of domain knowledge. As described in [30], the difficulty of the problem is changed based on the types of objects appeared in the problem. Therefore, to generate surface structure is regarded as finding appropriate objects and attributes. Table 3 introduces some systems that focus on generating the surface structure/problem sentence from given domain knowledge.

Table 3: Targets of existing problem generation systems for deriving surface structure

System	Target domain
[31]	SQL database query language
[32]	Real-life problem for self-assessment
[33]	Arithmetic

Martin and Mitrovic [31] applied problem generation mechanism into SQL database query language learning system. In the system, some constraints are described using the wildcard or variable. For example, in the constraint that checks "*WHERE clause in student solution contains name of table or attribute*", variables that represents *name written by student*, *table name*, and *attribute name* are used. Problem generation of this system is to prepare constraints whose variables are changed to available values. In order to set available values for each attribute, *instantiation constraint* is introduced. It restricts the value of the literals and checks if instantiated values are available. Example of the instantiation constraints is to ensure the literal in WHERE clause in student solution is valid attribute name of the table. Similarly, Le and Menzel [32] applied the constraint-based approach in the real-life problem[4]. It also embeds the function to check if the generated surface structure does not violate the hard constraints.

Kojima and colleagues [33] constructed problem generation system based on the differences between two problems called *episode*. Episode is formed by differences of two problems selected by the user; *base problem* and an example of *new problem*. Episode is regarded as a rule for changing the surface structure of the problem from one to another. New problem can be generated by applying episode to the existing problem. Generated problems are available if episode is applied to the correct type of problem.

3.4 Direction for Problem Generation in Educational Systems

There are three categories in introducing problems into the educational systems: 1) Problems are generated dynamically according to the teaching strategy in the educational system; 2) Problems are utilized by problem selection function in the educational system based on their metadata; 3) Problem generation framework is introduced into the educational system which supports the learner's problem generation activity.

As the first category, Le and Menzel [32] introduces the evaluation phase of learner's skills and personal characteristics in *the problem generation cycle*. Hirashima and colleagues [25] designed the system based on the concept: "*problems that simplify the original difficult problem is one of the most promising method to realize the help for a learner to solve a problem him/herself" [34]*. The system determines the solution structure of the new problem by simplifying that of the original one according to the learner's answer and generates three types of simplified problems[5].

For the second category, almost no practical system is proposed which embeds function for selecting generated problems. One reason is that most systems focus on generating only a specific part of the problem. For example, most systems do not consider steps of translating surface structure to the problem sentence or that of translating solution structure to the solution sentence. The other reason is that generated problems do not have metadata that represent their characteristics. In order to solve the second problem, Hirashima [30] proposed the metadata editor by which author is able to describe the metadata of problems that consist of objects, attributes, and values easily. However, the method for employing the metadata into adaptive support was not discussed.

Third category is one of the focusing topics in this area. Kojima and colleagues [33] introduced episode, which describes rule for changing the original problem to the new problem, to encourage learners to derive various types of problems in their problem

[4] The example of the problem is "*to arrange the family members around a table in the way that satisfy the important condition*", with conditions and attributes of family members, e.g. the interests, aversions and the favor of other family members, are provided.

[5] (I) Formulation partialized problem which simplifies the surface structure, (II) solution partialized problem which partitions solution structure, (III) solution specialized problem which specializes constraint structure.

generation activities. Yamamoto and colleagues [35] integrated the domain knowledge in physics into the system for supporting problem modification activities. In this system, learners make validate question sentence and also solve the problem that they generated. The system gives feedback for both generated question sentence and derived solution structure. Ishima and colleagues [27] also refer to the possibility of introducing the system into the learner's problem generation activity.

4 Conclusions

As introduced in this paper, many systems for QPG were developed. In the technical point of view, many systems are succeeded with their novel approaches. However, they are not widely accepted yet as a part of practical educational systems. To solve this situation, the way of deciding surface structure of questions/problems according to the strategies or the way of selecting already generated questions/problems are common topics for QPG.

References

[1] Rus, V. & Graesser, A. C. (2008). *Workshop report: The question generation shared task and evaluation challenge, Chapter 1*.
[2] Craig, S. D. & Dicheva D. (2009). *Proc. of the 2nd Workshop on Question Generation, AIED2009.*
[3] Boyer, K. E. & Piwek, P. (2010). *Proc. of the 3rd Workshop on Question Generation, AIED2010.*
[4] Dale, R. and White, M. (2007) (Eds.). *Position papers on the workshop on shared tasks and comparative evaluation in natural language generation*.
[5] Forăscu, C. & Drăghici, I. (2009). Question generation: Taxonomies and data. In [2], pp. 25-29.
[6] Kalady, S. et al. (2010). Natural language question generation using syntax and keywords. In [3], 1-10.
[7] Rus, V. et al. (2009), Building resources for an open task on question generation. In [2], pp. 48-52.
[8] Ali, H. et al. (2010). Automation of question generation from sentences. In [3], pp.58-67.
[9] Piwek, P. & Stoyanchev, S. (2010). Question generation in the CODA project. In [3], pp. 29-34.
[10] Pal, S. et al. (2010). QGSTEC system description - JUQGG: A rule-based approach. In [3], pp. 76-79.
[11] Boyer, K. E. et al (2009). An empirically-derived question taxonomy for task-oriented tutorial dialogue. In [2], pp. 9-16.
[12] Becker, L. et al (2010). What's next? Target concept identification and sequencing. In [3], pp. 35-44.
[13] Knight, K. & Marcu, D. (2000). Statistics-based summarization – step one: Sentence compression. *Proceedings of the 17th National Conference of the American Association for AI.*
[14] Heilman, M. & Smith, N. A. (2010). Extracting simplified statements for factual question generation. In [3], pp. 11-20.
[15] Punyakanok, V., et al. (2008). The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Journal of Computational Linguistics*, pp. 257-287.
[16] Ratinov, L. & Roth, D. (2009). Design Challenges and Misconceptions in Named Entity Recognition. Proceedings of the 13th Conference on Computational Natural Language Learning.
[17] Mannem, P. et al. (2010). Question generation from paragraphs at UPenn: QGSTEC system description. In [3], pp.84-91.
[18] Heilman, M. & Smith, N. A. (2009). Question generation via over-generating transformations and ranking. In [2], pp.30-37.
[19] Varga, A. & Le, A. H. (2010). A question generation system for the QGSTEC 2010 Task B. In [3], 80-83.
[20] Wyse, B. & Piwek, P. (2009). Generating questions from OpenLearn study units. In [2], pp. 66-73.
[21] Chen, W. et al. (2009). Generating questions automatically from Informational text. In [2], pp.17-24.
[22] Sneiders, E. (2002). Automated question answering using question templates that cover the conceptual model of the database. *Proc. of the 6th Int. Conf. on Applications of Natural Language to IS*, pp. 235-239.
[23] Mostow, J. and Chen, W. (2009). Generating instruction automatically for the reading strategy of self-questioning. *Proceeding of the 2009 Conference on Artificial Intelligence in Education*, pp.465-472.
[24] Hirashima, T. et al. (2006a). *ICCE 2006 WS Proc. of the Problem-Authoring, -Generation and –Posing in a Computer-Based Learning Environment*.
[25] Hirashima, T. et al. (2009). Problem Generation as Structure Simplification Following Problem-Solving Process. *ICCE 2009 Workshop Proceedings of the Modeling, Management and Generation of Problems/Questions in eLearning*, pp. 10-14.
[26] Hirashima, T. et al. (1994). An Indexing Framework for Adaptive Arrangement of Mechanics Problems for ITS. *IEICE Transaction on Information & System*, E77-D(1), pp.19-26.

[27] Ishima, N. et al. (2006). Developing Problem Representation, Knowledge Representation and Problem Solving System for Intelligent Educational System of High School Chemistry. In [24], pp.41-48.
[28] Matsuura, K. et al. (2006). The Problem Generation Module for Word Problem in the WBT System. In [24], pp.17-24.
[29] Kojiri, T. et al. (2006). Generation of Answers in Mathematical Exercises Based on Solution Network. *Journal of Information and Systems in Education*, Vol.4, No.1, pp.37-44.
[30] Hirashima, T. (2007). Differential Indexing as a Metadata Model of Exercise Problems. *ICCE 2007 WS Proc. of the Modeling, Management and Generation of Problems/Questions in eLearning.* pp.123-127.
[31] Martin, B. and Mitrovic, A. (2002). Automatic Problem Generation in Constraint-Based Tutors. *Proceedings of Intelligent Tutoring System 2002*, LNCS 2363, pp.388-398.
[32] Le, N.-T. & Menzel, W. (2006). Constraint-based Problem Generation for a Self-Assessment System. In [24], pp.33-40.
[33] Kojima, K. et al. (2009). Study on Support of Learning from Examples in Problem Posing as a Production Task. *Proceedings of 17th International Conference on Computers in Education*, pp.75-82.
[34] Polya, G.(1945): How to Solve It, *Princeton University Press*.
[35] Yamamoto, S. et al. (2009). An Approach to Promote Learning by Problem-Based Problem-Posing, *Proceedings of TELear2009*, poster.