A problem-solvers' assistant : towards scalable learning environments to foster planning ability

Itoh Kohji, Hasegawa Hiroyuki, Kaneko Hiroshi, Mihara Eisuke, Matsuda Hisayuki, Kakegawa Jun'Ichi, Fujii Masahiro, Itami Makoto

A problem-solvers' assistant : towards scalable learning environments to foster planning ability

ITOH Kohji, HASEGAWA Hiroyuki, KANEKO Hiroshi, MIHARA Eisuke, MATSUDA Hisayuki
KAKEGAWA Jun'ichi, FUJII Masahiro, ITAMI Makoto

Department of Applied Electronics, Tokyo University of Science
2641 Yamazaki, Noda-shi, 278-8510 JAPAN
Phone: +81 4 7124 1501 ext. 4206 Fax: +81 4 7122 9195
E-mail: itoh@te.noda.tus.ac.jp, hasegawa@itlb.te.noda.sut.ac.jp

Abstract

This paper first describes a computer-assisted environment for learning by problem-solving we have been developing for some years and now we are reengineering into a network distributed collaborative environment. According to the lessons learned in trying to obtain maximum scalability indispensable for training generalized ability of planning for solving problems, we renounce possibility of automatic plan generation and, instead, propose an authoring environment that provides the authors with on-the-shelf repertoires of reusable problem types with plans for solution as well as assistance of execution, both being embedded therein.

Keywords: computer-assisted learning by problem-solving, fostering ability of planning, scalability of material, reusable problem types.

Résumé

Cet exposé, en premier, rapporte sur un environnement assisté par ordinateur, pour l'apprentissage par solution des problèmes, que nous avons développé depuis quelques années et dont nous faisons remodelisation vers un environnement où les participants collaborent étant distribués sur le réseau électronique. Suivant les leçons tirées d'expérience d'essayer d'obtenir la maximum d'extensibilité des répertoires de problèmes indispensables à former la capacité générale d'organiser la processus à résoudre des problèmes, nous renonçons la possibilité de la production automatique des plans d'organisation et à sa place proposons un environnement pour les auteurs qui leurs donne répertoires, sur le rayon, des types de problèmes réutilisable avec menus des plans d'organisations à résoudre des problèmes aussi bien que l'aide d'exécution de la solution, tous encastrés dans les définitions des types de problèmes.

Mots-clés : apprentissage assisté par ordinateur, former la capacité d'organiser solution des problèmes, extensibilité de la matière, types de problèmes réutilisables.

Introduction

A number of intelligent tutoring systems and interactive learning environments have been developed and used [1 ; 2]. Some of them focused intelligent management of branch-indexed coursewares with traditional contents [3 ; 4]. Others provided for assistance of specific problem-solving tasks such as geometric proofs or algebraic manipulations, word problems or mechanics problems, and so on [5 ; 6 ; 7].

Taking into account the ever growing variation of problems we encounter in the process of fulfilling the role we play in either professional or everyday life, it seems impossible to know in advance everything necessary to solve problems. And the ever increasing accessible resources of information and knowledge available for solving specific problems tend to make us think we can attack those problems by retrieving necessary information in the resources on the fly. According to the pragmatic study of problem-solving, e.g. by G.Polya [8], Clancey [9], however, problems cannot be solved only by having a knowledge repertoire, but the ability of planning is required as to when and how to use what sort of knowledge in the process of solving problems, as tailored to the problem context concerned. Therefore, the final goal of learning nowadays should be acquisition not so much of specific knowledge collection as of the ability in general of planning how to solve problems. Very few systems, however, seem to have been implemented which assists planning problem-solving in pre-professional education. One of the rare examples was GUIDON [9] made by Clancey which assisted medical students of diagnosis of blood infectious diseases. From the constructivistic view points [10], we are aiming at providing the students in engineering education with a workbench environment for learning by problem-solving on which they can collaborate, discussing, in editing plan maps for solving system-given problems with assistance providing repertoires of problem types and embedded plans for solution.

We first describe our prototype system implementation as a case study into this direction, and comment on the results of an experiment. The top most concern in the study was investigation into scalability of such a system, because a sizable number of different problems with assistance are needed for the learners to acquire ability of solving new problems by generalization through experience of planning solution of related and differentiated problems.

Therefore, we then discuss reusable problem types and, renouncing possibility of automatic plan generation by the system because of the inability of the present computer systems of embodied interpretation of the context of the problems, propose an authoring environment which provide the authors with on-the-shelf repertoires of reusable problem types and plans for solution as well as assistance of execution being embedded therein. The authors can edit individual problem description and plan options for solving them combining the reusable problem types and augmenting problem-specific procedures if necessary. They should edit textual and diagrammatic interface that will bridge the abstraction of the reusable problem types and the concrete context of the problem.

A Problem-Solvers' Assistant

CAFEKS, abbreviation of "Computer-Assisted Free Exploration of Knowledge Structure", is a computer-assisted explorative environment for learning by problem-solving we are still in the midway point of developing [11 ; 12].

Now we describe first the authoring environment in which the authors should prepare a repertoire of problems as well as problem types to be used in the plans necessary in solving the problems as well as the plans to be used in solving the problems belonging to the problem types. Next we explain 5 features of assistance the learning environment provide learners with using the authored repertoires of problem and problem type definitions and their embedded plans for solution. Thirdly given is a brief description of implementation and finally commented on is extension to a collaborative learning environment now under way.

Authoring environment

The system provides for an authoring environment by which the author, maybe a domain-expert instructor, prepares a repertoire of problems with certain educational goals to be achieved in his domain. The same authoring system is used by the author or others, who might be domain knowledge experts, to add new entries to the repertoire of reusable problem types (simply "problem types" hereafter) combinations of which provide options of plans for solving the problems in the experts' domain. To each of the problem types are defined what is ASKED to be obtained under what conditions and what kind of data be GIVEN. Some of the problem types are strategic or meta-cognitive and others are certain ways of applying knowledge units. In each of the problem types the authors are requested to edit and embed plan options for solving problems belonging to the type by connecting the GIVENs with ASKEDs of problem types selected from the repertoire, thus establishing a usage tree of problem types including recursive usage. The problem type located at a leaf position of the tree without any embedded plan option is a certain way of using a knowledge unit and should be provided with an interactive process of assisting its execution. To each of such problems as do not belong to any problem type should be authored plan options for solving the specific problem by connecting the problem types selected from the repertoire with possible supplement of specific procedures. For each of the problems also required to be edited are descriptions of the feasible solution processes along feasible solution paths among those obtained by expanding the embedded plans. The descriptions are in XML format and the corresponding locations of the texts are tagged by the problem type ID used there and the tags will be used to highlight the locations when they are retrieved and displayed.

Learning environment: features of assistance

The system provides the learners with a workbench environment whose assistance is featured as follows:

1. First, the learners are assisted of comprehension of the system-given problem by describing what is ASKED, GIVEN what conditions and data, using the frameworks offered by the system.
2. Next, they are assisted of planning to solve the problem by selecting problem types from the menu and connecting GIVENs with ASKEDs each other to delineate "plan maps" along which the learners are encouraged to solve the problem by themselves.
3. They can also retrieve and refer to the solution process of such problems in the repertoire as are using, in the solution plan, the problem type they specified and the corresponding part of the solution process is highlighted.
4. When they deadlock or want to verify their solution, they can consult the system for the embedded menu of plans for solving the given problem as well as those for solving problem types comprised in the plans the learners have selected from the menu. They can thus proceed expanding plans they selected from the embedded menu or returning to self-planning on the way. When they come to a problem type which has no plan any more and we call "leaf" type, its GIVEN having been instantiated, they are assisted of executing solution of the instantiated problem type.
5. The system allows the learners to redo their selection of problem types in the self-planning phase and to redo plan selection and/or execution of leaf problem types in the consultation phase.

Throughout the system the design of reusable problem types are most critical and the levels of the learners should be taken into account in providing them with the more abstract problem types.

Implementation

The system is implemented in Java, using also XML and Prolog. It consists of CAFEKS package and domain packages, as shown in Figure 1.

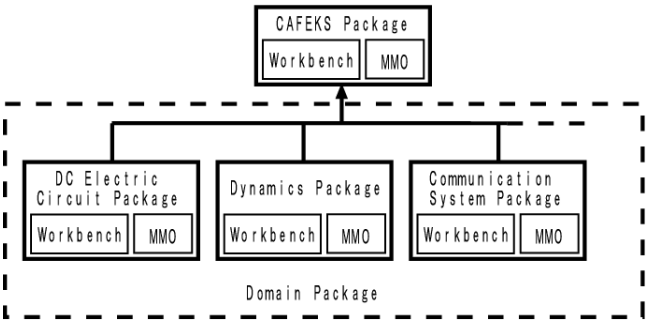

Figure 1. System Architecture

Each package consists of two packages. The one is "workbench package" whose classes are responsible for logical management and processing of the functions for both authoring and learning environment, and the other is "Media Metaphor Object(MMO)" package which takes control of the learner/author-computer interaction via GUI in relation to the workbench classes.

The CAFEKS package consists of core classes independent of domains for management of sessions, libraries of problems and problem types, as well as an abstract problem type class and a plan class from which domain specific problem types and plans are extended.

A domain package consists of specific problem type and plan classes dependent on the specific domain concerned.

We are developing a prototype system taking up direct current electric circuit theory as the sample domain. In the following described is the implementation of the 5 features of assistance mentioned in section **Learning environment : features of assistance**.

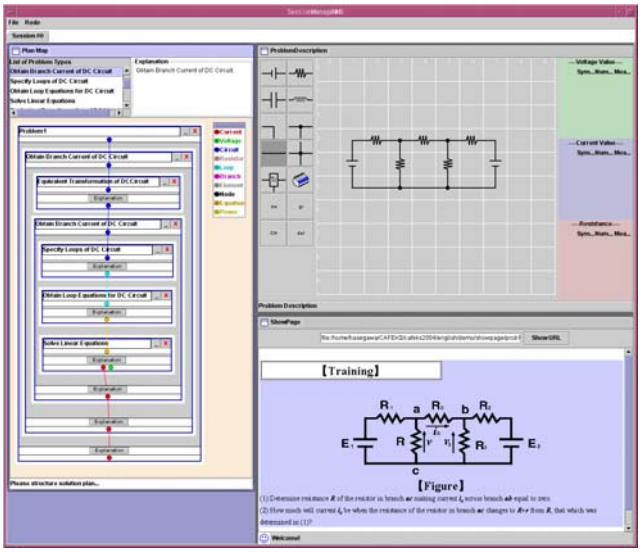

Figure 2. A Screen Snap Shot of Planning Assistance

For feature (1), a laboratory-made tiling editor was implemented in the DC circuit analysis problem type class in order for the learners to construct circuits and specify symbols designating resistances, source voltages or specifying closed loops, etc. For features (2) and (4), the "plan map" editor was implemented in the plan class with a function to save the edited plans into files by serialization in XML and to load leaner-specified plans by deserialization. Also implemented for feature (4) was assistance of executing a leaf type problem which prompts the learners to input their solutions and gives them feedback of verification. Fig.2 shows a screen snap shot of planning / execution assistance. At the right-bottom is given the training problem. At the top-left is the menu of the built-in problem types. On the right hand upper area, the learner can draw the circuit with specification of symbols or values of currents, voltages and/or resistances. On the left hand middle area, the learner can edit her/his plan, selecting problem types from the menu and connecting or expanding plans embedded in the problem types. You can see on the plan map that, before the time the screen shot was taken, in planning phase, the problem type "Obtain Branch Current of DC Circuit" was selected, and the plan "Using Equivalent Transformation" was adopted consisting of the problem type "Equivalent Transformation of DC Circuit" and recursive call of the problem type "Obtain Branch Current of DC Circuit". Then the latter was expanded and the plan "Using loop equation" was selected consisting of the problem types "Specify Loops of DC Circuit", "Obtain Loop Equations for DC Circuit" and "Solve Linear Equations".

For feature (3), solution process was tentatively described in TEX format for each of the problems in the repertoire and tags were written in the TEX comments designating the problem types used in the tagged locations of the solution process.

The TEX files were parsed by a CGI script and the tags were used to search for such problems and locations in the description of the solution process as using the learner-specified problem types and to highlight those locations in displaying the description.

Now an editor for the solution process descriptions in XML format is being developed.

For feature (5) the system allows the users to redo executing or planning any portion of the plan map using a kind of truth maintenance technique. In order to enable redoing execution, the status of the result of executing an executable sub-problem is recorded by serialization. In redoing, the session before redoing is saved and a new problem-solving session is created using the status record, thus allowing the learners to recapture and resume the previous sessions.

Towards Collaborative Learning Environment

We are now reengineering the stand-alone system into a collaborative learning environment on an intranet. Using Java RMI (Remote Method Invocation) mechanism, we set up bidirectional channels between the clients via a server controlling the communication in order to realize a collaborative workbench environment and a voice chat system with annotation by voice recognition for use in reflection. As for the collaborative workbench with MMO environment, the learner client systems are run in synchronism, while each client has its own private workbench with MMO and the learner can temporally replace the status of the private workbench with that of the collaborative one and privately extend the collaborative work. And, conversely, s/he can propose and upload the result of her/his private work to the collaborative workbench. In the collaborative environment, the method invoked by a learner's action approved by the members via the voice chat system, is executed at all of the clients. In prospective of realizing the environment on the internet in the future, we adopted a scheme in which a method invocation is encoded at the transmitter side into a string, being sent to and interpreted at the receiver side.

A Preliminary Experiment for Evaluation

We made a preliminary experiment for evaluation of the system concept to see if feature (3) of assistance, although implemented following the tentative specification mentioned in section **Implementation**, works for learners to be motivated to use newly learned plans in solving new problems.

The target domain was direct current electrical circuit theory.

The 22 reusable problem types were provided among which many were used in the solution process of the 10 sample problems in the repertoire.

Fig.3 shows a snap shot of plan-retrieved solution process description taken in the experiment. 12 subjects, all being the 3rd year university students of electrical engineering course, were divided, according to the pretest, into two groups with similar distribution of grade points. They were given training problems difficult to solve without using plans suggested : deletion of the to-be-zero current branch and use of the compensating voltage source. While studying the training problems, the 6 control group students were allowed only to look paper handouts of sample problems with solution and the 6 experiment group students used the computer system being allowed to retrieve such sample problems and locations in the solution process where plans and problem types they specified were used.

In the post test, a problem closely related with the training problem was given. 5 out of 6 students of the experiment group tried to use the plans suggested in the training, whereas only 3 out of 6 students of the control group tried to use them.

This difference between the groups and the answers of an enquete and the interviews after the experiment at least suggest the favorable effect of plan-focused retrieval of the solution process in motivating and encouraging use of the newly learned knowledge.

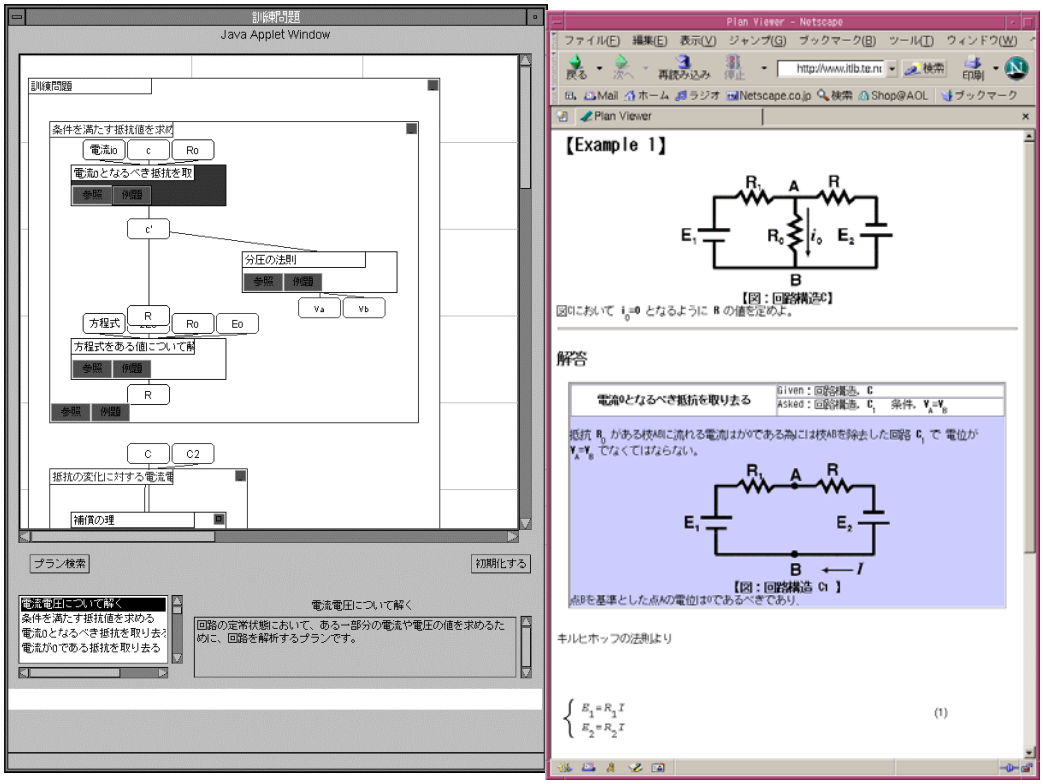

Figure 3. A Snap Shot of Plan-Retrieved Solution Process Description

Reusable Problem Types and Heuristics : Lessons Learned

We searched for structures of plans reusable in as large classes of problems as possible and found that it was possible to classify reusable types of problems according to the domain-independent categories of what is asked and types of given conditions, and to write down their plans consisting of connections of domain-independent problem types.

These problem types the top level plans comprise can subsume domain-dependent problem types which have plans consisting of domain-dependent problem types.

The domain-dependent hierarchies of problem types and plans for solving problems of the type depend on the problem domains, and problem types comprised in the plans finally goes as far as to those representing certain ways of using domain-specific knowledge units not made to have plans any more.

From the viewpoint of how to discover plans for solving mathematics problems, G.Polya studied useful heuristics [8]. Each of the heuristics actually is useful in certain phases of solving very broad types of problems in any domain in which mathematical formulation is used. Polya's heuristics should be for any level of problem types in the plan-problem type hierarchy.

Plans and heuristics, however, can never be directly transferred to the learners. The only way of training the learners to be able to use them by themselves is through experience of solving problems with proper heuristics and plans timely suggested [9].

Accordingly, systems are more demanded which provide for assistance of planning to solve problems and to use heuristics.

As was stated in **Introduction**, we are aiming at providing the students in engineering education with a workbench environment for learning by problem-solving on which they can collaborate in editing plan maps for solving system-given problems with assistance providing repertoires of problem types and embedded plans for solution.

In such an environment, a sizable number of variations of problems are needed in order for the students to inductively capture, between different problems, common features enabling them to apply the same or similar plans and heuristics and also to differentiate usage of plans and heuristics according to the problems concerned.

Therefore in developing systems that assist learning by problem-solving, scalability in providing problems with assistance is a very important issue.

Ideally, we might let the system have capability of autonomously producing problems and autonomously developing plans for solving them with advice of heuristics. Both projects, it seems, however, be doomed to fail because we (with other AI researchers presumably) found the following:

1. Preparation of semantic frameworks necessary for the individual problem description to produce linguistic expressions for conveying meaning to the students (and vice versa, from linguistic expressions to semantic framework expressions) is often ad hoc and the conversion requires capturing context whose bodily origin [13] prohibits making explicit rules of conversion as well as universal semantic frameworks. And therefore, it is impossible for the system to provide such frameworks and conversion rules in advance of problem generation: e.g. word problems : e.g. linguistic expressions of problems on electromagnetics or 3-dimensional vector analysis can never be complete, and require a lot of default interpretation in converting them into a coordinate system description which is the only logical framework for such problems: e.g. in piping unix commands, grep | sort results in the same as sort | grep; any knowledge explaining why this is so should be ad hoc.

2. Selection of problem types and knowledge units to be applied in a plan largely depends on the context of the problem and often needed is to introduce pertinent assumptions and verify afterwards: e.g. a moving body could be deemed a point mass or should be treated as a rigid body: e.g. the mass of the pulley could be neglected or not in the apparatus of Atwood: e.g. applying static or dynamic friction coefficient or neither, assuming the mass being stationary or moving on or off the slope.
3. Use of the problem types and knowledge units must be adapted to the problem context [14] by way of procedures specific to the problem: e.g. ways of dividing a given 3D figure into primitive figures or introducing certain augmented figures for obtaining its volume could not be generalized as rules: e.g. it seems impossible to work out rules on introduction of pertinent auxiliary figures in geometric proofs or construction problems.

In consequence, we are forced manually to prepare repertoires of problems aiming at certain learning goals and manually to provide the problems with plan options for assisting solving problems as well as advice of heuristics to assist discovering plans. Nevertheless, in our experience, it seems possible to design such a repertoire of generalized reusable problem types with varied domain-dependency and ways of using domain-dependent knowledge units as could actually solve, via combination, problems found in the textbooks of the mathematically-oriented domains of higher education. And planning level semantic frameworks for producing expressions necessary and sufficient to discriminating applicability of those problem types could be devised. Moreover, such an abstract level of semantic framework is necessary as discriminating applicability of the problem types and knowledge units appearing in the plan. The linguistic level of semantics is never made use of. Linking the abstract level semantics with students' comprehension is made via template-grammatical and text-diagrammatical interfaces embedded in the individual problems. In conclusion, being provided domain-dependent library of reusable problem types whose GIVENs and ASKEDs as well as plans for solution are described using such semantics as necessary and sufficient for determining their applicability, it will be possible to provide domain-expert instructors with an environment that assists authoring the problem descriptions, plans and heuristics for solution.

Examples of Reusable Problem Types and Plans

Problem-solving is defined in most general terms as an activity of selecting from among a given set S of candidates all or some of such members (ASKED) as satisfy given conditions (GIVEN). Set S may be finite, countablly infinite, or neither. In case S is finite, a general plan of one by one examination could be used.

List of Typical Problem Types:

CATs (Categories) in the following are:
entity/field/geometric/symbolic system

To be observed or to be constrained are:
category/attributal values/structures/relations/
time-space locations (or movement)/ operations (or construction)

Objects satisfying the constraints are variables, while objects as observed are not.

From GIVEN excluded are available knowledge units which could be freely used.

1. CAT : proof problem:
ASKED: proof processes. GIVEN: propositions to be proved
2. CAT : attributal value constraining problem:
ASKED: attributal values, GIVEN: constraints
3. CAT : relation constraining problem
ASKED: relations. GIVEN: constraints
4. CAT : structure design problem
ASKED: structures. GIVEN: constraints, min/maximization of attributal values
5. CAT : operation design problem
ASKED: series of operations. GIVEN: constraints, min/maximization of attributal values
6. CAT : construction design problem
ASKED: series of construction operations. GIVEN: constraints, min/ maximization of attributal values
7. CAT : diagnostic problem
ASKED: category/structure/relation. GIVEN: observations
8. CAT : knowledge discovery problem
ASKED: proposition (or knowledge). GIVEN: observations

Once solved, the result makes a knowledge unit. Conversely from any knowledge unit, problems can be created.

An example of geometric attributal value constraining problem

ASKED: attributal value.
GIVEN :constraints: category of geometric figure.
Domain Problem:

Obtain the volume of a rectangular frustum given the height and the lengths of the upper and lower edges.

Generic Plan:

Discover and use the relation between the attributal values of the parts and that of the whole of the geometric figure. If needed, introduce auxiliary figures.

Domain Plan:

prob: Introduce 2 auxiliary rectangular cones.

prob : Calculate the volume of the frustum as difference of that of the larger cone and that of the smaller cone.

prob: Assume the unknown height of the smaller cone.

prob: Obtain the edge ratio equation with the triangles in the similarity position

An example of entity/field attributal value constraining problem

ASKED: attributal value.
GIVEN: constraints: 2 entity system time-space location in a field.
Domain Problem: 2 bodies b1, b2 fall from a tower in the gravity field. b1 falls from the top of the tower, b2 falls from height b[m] down from the top when b1 has fallen a[m] down from the top. Obtain the height of the tower and the time required for b2 to fall to the ground.
Generic Plan:

Discover and use laws of the 2 entities B1 and B2's movements.

Divide time into intervals with simple movements of B1, and B2.

Solve for the movements of the entities, B1, B2 in each interval.

Represent constraints.

Solve for the ASKED.

Domain plan:

prob: Usage of Newton's law of motion of point mass.
prob: Divide time into 2 intervals, assume the border time, and the border states.
"single body falling interval and 2 bodies falling interval"
"border time t1 and border velocity v1 of b1"
prob: Solve for the movements in the first interval.
prob: Represent constraints.
Prob: Obtain values for the border.
prob: Solve for the movements in the second interval.
prob: Represent constraints.
prob: Solve for the ASKED.

An example of entity/field structure design problem

ASKED : structure of an entity.
GIVEN : partial structure, attributal values, minimize an evaluation function.
Domain Problem:
Design a coded communication system consisting of an encoder-decoder pair and a digital channel, the digital channel being a digital modem channel consisting of a modulator-demodulator pair and a physical channel, the physical channel consisting of a transmitter-receiver pair and a radio propagation media. The information bit rate and the BER quality are given as requirement and minimize combined transmitter-power and bandwidth evaluation function.
Generic Plan:
Assume a structure ST with candidate parts PT hierarchy.
Define constraining variables with which parts PT are to be connected .
For each of the candidate parts PT, solve the relation constraining problem as to the relation between those variables and, if necessary, record the results in a database.
Solve the attributal value constraining problem as to the value of the evaluation function with the given constraining conditions.
Change ST replacing PT and repeat the above to find optimum value of the evaluation function.
Domain Plan:
prob: Assume a coded communication system hierarchy consisting of an encoder, a digital channel, a decoder; the digital channel consisting of a modulator, channel, demodulator; the channel consisting of a transmitter, propagation media and a receiver.
prob: Define constraining variables; bit transmission rate(BTR), bit error rate(BER), band width(B), signal-to-noise ratio (SNR), transmitter power(TXP), receiver noise figure(NF), attenuation on propagation (ATT), etc.
prob: For each hierarchy and each scheme, solve for the relation between outer BTR, BER and inner BTR, BER, or between B,SNR and BER, or between TXP, ATT, NF and SNR, etc.
porb: Minimize combined TXP and B
evaluation function with the information
bit rate and the BER quality being given.
prob: Replace the codec/modem schemes and repeat the above to find the optimum.

Conclusion

We first described CAFEKS, a computer-assisted explorative environment for learning by problem-solving we are developing and extending to a collaborative environment. According to the lessons learned in trying to obtain scalability indispensable for assisting the learners to acquire generalized ability of planning for solving problems, we discussed reusable problem types and, renouncing possibility of automatic plan generation by the system because of our inability of devising universal semantic expression frameworks for the computer and the inability of the computer of interpreting the context of the problems, we proposed an authoring environment which provides the authors with on-the-shelf repertoires of reusable problem types, and plans for solution as well as assistance of execution, both being embedded in the problem types. Using these reusable problem types and their semantics in describing plans for solving individual problems and manually filling the gaps between the problem-specific semantics of bodily origin and the planning level abstract semantics will provide the learners with learning environments scalable in the range of problems covered by the system sufficient for assisting the learners to acquire ability of planning.

References

[1] Wenger E. 1987: *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann.

[2] Forbus K.D., Feltovich P.J. 2001: *Smart Machines in Education*, AAAI Press/The MIT Press

[3] Kiyam,M.,Ishiuchi,S.,Ikeda,K.,Tsyjimot,M.Fukuhara,Y. 1997 : Methods for the Web-Based Intelligent CAI System CALAT and its Application to Telecommunication Service, *Proc. AAAI-97*,Providence RI

[4] Ayoro,L.,Dicheva,D. ,Velev,I. 2001 :A Concept Based Approach to Support Learning in Web-Based Course Environment, *AIED01*, pp.1-12. IOS Press

[5] Anderson,J.R.et al. 1985 :The Geometry Tutor, *Proc.IJCAI*, LosAgeles pp.1-7

[6] Hirashima,T.,Horiguchi,T.,Kashihara,A.,Toyoda,J. 1998 : Error-Based Simulation fro Error Visualizationand Its Management, International, *The International Journal of AIED*, vol.9,no.1-2,pp.17-31

[7] VanLehn.K. 2000 : Andes: A Coached Problem Solving Environment for Physics, *ITS 2000*, pp.133-142

[8] Polya,G. 1962 : *Mathematical Discovery*, Vol.1-2, John Wiley & Sons

[9] Clancey,W.J. 1987 : *A Knowledge-Based Tutoring: The GUIDON Program*, MIT Press

[10] Jonassen,D.,Mayes,T.,McAleese,R. 1992 :A Manifesto for a Constructivist Approach to Uses of Technology in High Education, in Duffy,T.M.,Lowyck,J.,Jonnassen, pp.231-247, D.H.(Eds.): *Designing Environments for Constructive Learning*, Spriner-Verlag

[11]Fujihira M., Kawamura T., Kawakami K., Itami M., Itoh K. 1999 : CAFEKS: An Atchitecture for Evolutional Development of Interactive Learning Environmnt with Coached Problem-Solving, *Proc. ICCE99*, vol.1, pp.915-922

[12] Kawkami K., Watanabe T., Tateno M.,Tabaru Y., Itami M., Itoh K. 2001 : Learning by Problem-Solving Assisted of Planning and Retrieval of Sample Problems with Solutions Indexed by Structured Solution Plan, *Japan Journal of Educational Technology*,vol.25,no.2

[13] Lakoff,G. 1987 : *Women, Fire, and Dangerous Things*, The University of Chicago Press

[14] Dieter 2000