To appear in the Proceedings of Conference on Creating, Connecting and Collaborating through Computing (C5), Kyoto, Japan, January, 2003.

A Conceptual Framework for Learning Experience Design

Kumiyo Nakakoji[1,2], Kazuaki Yamada[1], Yasuhiro Yamamoto[2,1] and Masayasu Morita[3,1]
[1]Research Center for Advanced Science and Technology, University of Tokyo, Japan
[2]PRESTO, JST, Japan
[3]Eduplanet, Japan

RCAST, University of Tokyo
4-6-1 Komaba, Meguro, Tokyo, 153-8904, Japan
{kumiyo, yamada, yxy, morita}@kid.rcast.u-tokyo.ac.jp

Abstract

This paper describes our approach of utilizing a computational environment to apply a learner's learning process as a first class object. With this approach, the learner's learning process can be used as communicative media among learners and teachers or learning content object providers (LCOPs). We argue that the design of a computer system is as equally crucial as the selection of an instructional method. The selection of which instructional method to apply and the design of a computer system together form a learning environment, which determines a learner's learning experience. This paper first discusses the learning experience design approach for the development of computer technology in general. Then we present a particular design of learning experience, where learners and LCOPs can communicate with each other via learning materials in the computational environment. We have developed an architectural framework for the learning experience, and applied the framework to three different educational settings.

1. Introduction

An important goal in an educational setting has been and will be that [18]:

- a teacher, or a learning content object provider (LCOP) in general, would like to understand how learners learn the object of concern in order to develop a better way of teaching them; and
- a learner would like to understand what learning style works for him/her and what is his/her most appropriate way of learning.

This goal has been achieved primarily through two methods; by conducting examinations, and by having more "eyes" on each learner. This has been true for most of educational situations regardless of what instructional methods they apply.

For instance, in a traditional face-to-face classroom, teachers use examinations as a learning measurement to gain an understanding of how each student has learned from their lectures. At the same time, each student has an opportunity to reflect on what part of the subject topics he/she is good (and not good) from their examination scores, thereby requiring further study.

In a technology-integrated learning approach, such as using an ITS (Intelligent Tutoring System), the system constructs a user model based on the learner's performance. By enabling a "one-on-one tutoring" style [2], the system then automatically adapts its teaching style to the user based on the constructed user model. With this approach, instead of a learner understanding his/her own learning style and choosing an appropriate way of learning, the system, using AI techniques, infers what is "more appropriate" for the learner and automatically adapts the way of teaching for the learner.

With the "Logo-as-Latin Paradigm" [11], although its teacher-learner role division becomes obscure by dealing with the issue of instructional transfer [10], program execution has played a role of examination. By looking at the execution of a program that a learner constructed, the learner as well as instructors can gain an understanding of what aspects the learner has and has not learned.

Thus, examinations are serving as communicative media between teachers and learners. The problem of this examination-based communication, however, is that examinations are merely a series of snapshot of a learning processes. They represent discrete representations of the learner's accumulated knowledge up to the point when the learner takes each examination. These series of examinations cannot capture such situations as when and how a learner gets confused, or when a learner has given up if the learner does not take an examination.

In this paper, we describe our approach of using a computational environment to use a learning process as a first class object. With this approach, the learner's learning process itself can be used as communicative media among learning peers and teachers, or LCOPs.

In doing so, we argue that introducing computers to current ways of teaching and learning does not necessarily either fully improve the current ways of teaching and learning or fully utilizing the computational power. While computers have been regarded as one type of technology that can be "used" in a variety of instructional methods, our view is that the design of a computer system is as equally crucial as the selection of an instructional method. The selection of which instructional methods to apply and the design of computer systems together form a learning environment [12], which determines a learner's learning experience.

This paper consists of two parts. The first part starts with a discussion on the problem of current ways of using computer systems in educational settings, and stresses the importance of learning-experience-design approach. The discussion is applicable to a wide range of approaches taken in supporting learning and teaching by using computer systems.

The second part describes our approach of a specific design of learning experience that uses learners' learning processes as communicative media among learners and teachers (or LCOPs) in order to achieve the goal described above. Section 4 presents a component architecture we have developed that enables the learning experience. Section 5 shows a case study of applying the framework to three different learning settings: E-Learning, classroom learning, and open-source software development.

2. The Use of Computer Systems in Educational Settings

A notable success of using computer technology in education has been the use of Logo, and other programming languages for kids [17][20][23]. They use computers as tools and materials that learners work on.

This paper, in contrast, focuses on a much wider context of the computer use in education. Many of CSCL techniques, such as E-Learning [21], distant learning [1] or Web-Based Training, use computer systems not as tools to work with, but as media for instruction materials, communication media and as shared workspaces.

2.1. How the Computers are Currently Viewed

Current ways of using computer systems in education are mostly done by regarding them as one type of technologies. Just like they had introduced film, radio, and television into classrooms with varying degrees of success [3], educators have started using computer systems in the context of their traditional ways of teaching and learning.

Many uses of computer systems in educational settings, therefore, try to use existing computer tools and systems on an "as is" basis, exploring where they can use this technology in their current practice.

Let us take E-Learning, for instance [13]. Initially, university classes have started putting their syllabi on the Web so that students can have a look at them anytime anywhere. Following this, they have started putting lecture notes on the Web. Then, examinations are put on the Web so that students can take them remotely. Upon recognizing the importance of communication among classmates and with teachers, electronic bulletin boards are provided. Nowadays, Q&A systems are provided so that students can have more focused correspondence with teachers.

This seemingly evolutionary development of computer usage, however, has not necessarily improved the quality of learning processes. We see two dependent factors involved in this. First, traditional ways of doing and representations do not necessarily work best in the computer technology. For instance, lectures notes that have been developed and used in a more "traditional" setting have been found not appropriate for the Web browsing due to the lack of appropriate design and information structure.

Second, system developers and educators tend to overlook how much impact technical details of a computer system have on a learning process. For instance, having an electronic bulletin board system (BBS) for a class involves a number of design options. Having a *public* BBS and having a *private* BBS have very different consequences because people see them as completely different media. Visual designs, such as the use of font and color, also affect the feeling for the system [24]. Failure to attend to such details may result in unexpected, unwelcoming consequences.

2.2. The Possible Impact of Computer Systems

There are two issues behind the problem with such approaches.

The first issue is that different from previous technologies introduced, such as film, radio, or television, computer technology can be "designed." They can be designed so that they will fit in a certain use context. At the same time, because technical details of computer systems do affect people's cognitive as well as social processes, computer systems need to be designed carefully so that they will result in

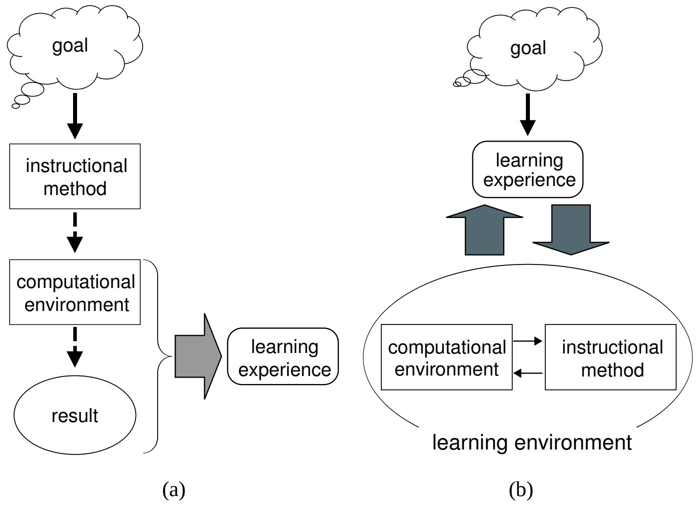

Figure 1. A computational environment and the learning experience: (a) a common view, and (b) our view

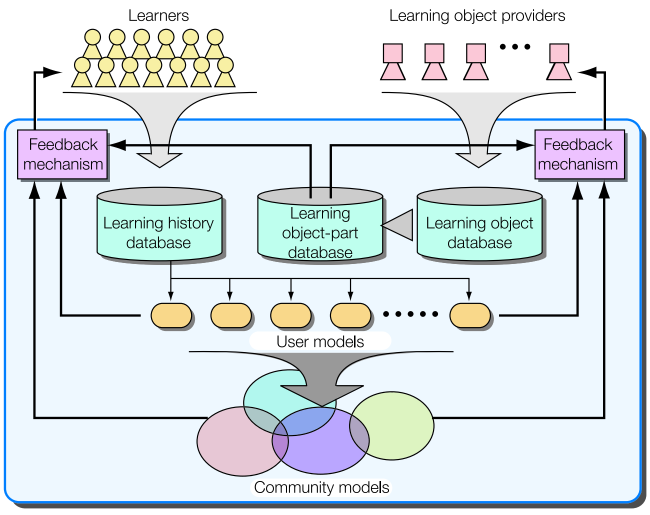

Figure 2. An Architectural Framework for the Designed Learning Experience

desirable consequences.

The second issue is even more critical. Existing approaches try to use computer systems in the current ways of teaching and learning; that is, within the tradition. Because of its inherent power, however, the use of computer systems will often result in transcending the tradition. The use of computer systems changes the way people do their work [4], and learn [5].

In order to transcend the tradition in a desirable manner, therefore, one must *design* computational technology so that the use of the technology will result in desirable changes. By computational technology, we include software systems, the Internet, as well as hardware.

The current situation of not-too-successful computer usage in education has been attributed to the system developer side by failing in fully appreciating the expectations and requirements of classroom practitioners [3]. However, we argue that even if systems are designed exactly as classroom practitioners expected, such systems are likely to fail because the practitioners often do not envision the future and they do not talk about the transcendence, but only about the tradition [16]. This is not due to the lack of their imagination or creativity, but due to the lack of understanding of what computer technology can do and will do.

Practitioners know the tradition. Computer developers and engineers know what computer technologies can do and cannot do. Together they need to envision the transcendence and design computer technology, requiring a collective creative process [14].

3. Learning Experience Design

As stated above, different technical details of computer technologies have different consequences on people's cognitive and social processes. This is so because such systems lead people to have different *experiences.*

Traditionally, in learning and education research, learning experience has been regarded as a consequence of learning. Fig.1(a) illustrates this view. As in a typical instructional design [12], starting with a learning goal, an instruction method is determined, and learner learns based on this method, followed by an evaluation of the learning result. The learner's learning experience, then, is reflected as a consequence of applying the instructional method.

In our learning-experience design approach, the order is reversed. We introduce computational technology as a first class object in the process. We start with the design of learning experience. Instruction method is selected and computational technologies are designed in accordance with the designed learning experience.

We view computational technology as a first class object in an instructional design. Which instructional method to use affects the design of computer systems. What kind of computer systems to use affects how the instructional method is implemented. The selection of an instructional method, and the design of computer systems, form an educational environment [6]. This educational environment determines a learner's experience (see Fig.1(b)). Note that we use the term "learning experience" different from "experiential learning" [8]. Experiential learning is a learning theory that sees knowledge as a transformation of experience.

In contrast, while we do not oppose this theory, what we mean by "learning experience" is an experience of a learning process.

4 Learning Experience Design for Broadening the Communication Channel

We now describe our design of a learning experience to achieve the goal stated in Section 1: (1) that teachers or learning content object providers want to understand how learners learn the object of concern in order to develop better ways of teaching them; and (2) that learners want to understand what their learning styles are and what ways of learning are most appropriate for themselves.

The designed learning experience, roughly speaking, enables learners to communicate with teachers or LCOPs via learning materials. In the same manner, the teachers and LCOPs are allowed to communicate with the learners via how learning materials have been used.

4.1. An Architectural Framework for the Learning Experience Design

In order to produce the above learning experience, we have taken an approach to design a computational environment where each learner's learning process is captured. The captured learning processes then become available both for learners themselves and for teachers or learning content object providers.

Fig.2 shows an architectural framework for the learning experience design. The framework takes *learners* on one hand, and *learning object providers* on the other hand.

In this framework, teachers, instructors, or authors of learning content objects are all viewed as learning content object providers (LCOPs). The separation of learners and LCOPs does not mean to preclude a situation where learners become LCOPs or vice versa; the same person may take the role of a learner or of a LCOP.

User models and community models are generated based on (1) the history of learners' learning processes and (2) learning content objects provided by LCOPs. Those models are then used to give feedback and task-relevant information to the both types of stakeholders. Using this framework, each learner can communicate with the LCOPs as well as with their learning peers throughout the learning process.

4.2. Components of the Architectural Framework

The architectural framework consists of the following six components:

- *a learning history database* that stores histories of each learner's learning processes. This component keeps track of what learning objects each learner looks at how long, and what and how he/she performs with the learning object.
- *a learning object database* that stores learning content objects authored by learning content object providers.
- *a learning object-part database* that stores parts of learning objects. This component segments each learning object into object parts (pieces) so that a new learning object that is more suitable to a learner can be derived by reassembling parts from multiple learning objects.
- a set of *user models* each of which characterizes each learner with indices, such as test scores and learning time. A user model is constructed by applying data-mining techniques to the raw data stored in the learning history database.
- a set of *community models* that classify learners by their similarities based on their user models. A community model is constructed by applying information filtering and information clustering techniques to the learning history database and learning object part database.
- *feedback mechanisms* give timely and task-relevant feedback to both learners and LCOPs based on the user models, community models, and learning object part database.

The essential aspect of the framework resides in the feedback mechanisms, broadening a communication channel between learners and LCOPs. The framework gives feedback to learners on how their learning have been progressed and how their learning peers have been doing in comparison. The framework gives feedback to LCOPs on how parts of each learning content object have been used by learners. This would then allow the LCOPs to better focus on which parts of a learning content object to be revise and modified, and what parts are missing in the current learning content objects.

The rest of this section describes user models, community models, and feedback mechanisms in detail.

4.3. User Models

Looking back to his/her own learning processes allows the learner to reflect on how his/her learning has been proceeded. Looking at his/her peers' learning processes allows him/her to refer to other learning styles, potentially better ones that the learner currently employs.

LCOPs, on the other hand, become able to keep track of how learning materials, curricula, and tools they have provided been used by each learner by having information on each learner's learning process.

A learner's learning process is implied by a *log* of data stored in a learning history database. However, each piece of data stored in the database, such as scores of each questions or time taken to answer a question, is not informative in terms of characterizing one's learning process.

User models are used to transform such a raw, large volume of log data into a meaningful chunk of information representing a learner's learning process.

In our current implementation, we use feature vectors to represent a user model. Feature vectors include:

- each learner's properties, such as sex, age, and previous learning histories,
- log data of test scores and time taken to solve each question,
- comparative data with other learners, such as average scores and standard deviation, and
- the types and frequencies of learning materials and tools used by each learner.

The log data is kept updated as a learner's learning proceeds. The comparative data is calculated periodically from the log data of all learners.

4.4. Community Models

Our architectural framework (Fig.2) presumes a model where a varying number of a variety of learners with different levels of skills and knowledge exist.

Community models are used to categorize learners into groups so that each group contains learners with "similar" learning styles.

Community models are generated by applying social filtering [22] and collaborative filtering [9] mechanisms to the *learning history database* and *learning object database*.

There are three issues in the generation of the community models.

1. Representation of input data for filtering mechanisms. Feature vectors used in *user models* have a large number of dimensions. Since there is a wide variation among learning speed and time taken to solve each question, such vectors are pretty sparse. In addition, the feature vectors include both numeric data (such as test scores, and time) and non-numeric data (such as professions). These characteristics of the feature vectors make it difficult to apply regular clustering methods.

   It is necessary therefore to apply the singular value decomposition or principal component analysis techniques to decrease the number of dimensions of the feature vectors as a pre-processing procedures before applying social and collaborative filtering techniques.

2. Clustering methods. Since we cannot predict how many groups would exist in community models, the use of decision trees and self-organization maps (SOM) is necessary as clustering methods.

3. Representation of community models. As illustrated in Fig.2, each community model is not mutually exclusive; a learner may belong to multiple community models. Also, granularity of a community varies, and a community (e.g., one representing "those who successfully solved a particular problem" may include another community (e.g., one representing "those who successfully solved the problem within a certain amount of time").

   Thus, we must use such techniques as a decision tree or Star Coordinates [7] so that we have control over the determination of the granularity of communities.

4.5. Feedback Mechanisms

A feedback mechanism consists primarily of (1) a mechanism to integrate user models, community models, and learning materials, and (2) a visualization mechanism to visualize the result of the integration.

By simply presenting generated user models and community models to learners and LCOPs would not bring about the designed learning experience. We use learning materials as a way to represent the community models in a more "meaningful" manner.

For learners, instead of presenting a list of names of their learning peers that belong to the same *community model*, the system must present customized learning materials dynamically composed of what those learning peers have been using. In the same manner, for LCOPs, instead of showing how each learner has been using learning materials, the system must present how each part of the learning materials have been used by which *community models*.

In summary, instead of giving learners and LCOPs a log of data, the architecture makes a deliberate effort of transforming the log into meaningful information to the both stakeholders, which will then allow learners and LCOPs to have a learning experience where they can communicate with each other via the computer environment. This is the essence of our designed learning-experience.

5. Applications to Three Settings

The framework described in the previous section does not presuppose any specific instructional methods. This

Table 1. Comparison of the Three Settings

	E-Learning	Classroom Learning	Open-Source Development
goal	gain English skills	transfer/construct knowledge	obtain necessary source code become core members
learner	consumers	students	end-users, new-comers
LCOP	learning objecte developers Web designers system managers	teachers	core members
learning term	fixed	fixed	open
learning time	open	fixed	open
learning pace set by	learners	teachers	learners
number of learners	100's to 1000's	10's	vary
roles	stable	stable	dynamic
environment	Web	classroom	Web
communication	Email, online-chat, BBS	face-to-face	Email, telephone
identity	little important	important	may be important
community development	less emphasized	encouraged	emerging

section describes a case study of using the architectural framework for three different educational settings, where different technical concerns have emerged: (1) commercially-oriented E-Learning, (2) classroom learning, and (3) open-source community development.

5.1. An Overview of Three Learning Settings

There are several aspects to note in comparing the characteristics of the three settings. Table 1 summarizes the comparison.

The relationship between learners and LCOPs remain the same for the E-Learning and Classroom settings because they both use a fixed learning term: those who are learners remain as learners within the same subject. In contrast, with no explicit learning term set, open-source community has more flexible role divisions. Over a long period of time, peripheral participants of an open-source community may learn and evolve into a more central role becoming LCOPs [15][25].

The E-Learning and open-source community settings have similar degrees of freedom in how much control each learner has over the learning process. The classroom setting, on the other hand, imposes relatively strict pacing for learners.

Since maintaining "identity" is important for learners in the classroom learning and open-source community, it makes it easier for those settings to develop and evolve communities. In contrast, with E-Learning, many learners tend to remain anonymous to other learners and even to LCOPs, and thus making it more difficult to formulate communities.

In different learning settings, learners and LCOPs have different purposes. Thus, different feedback mechanisms are necessary for the three learning settings.

The remainder of this section describes how the architectural framework described above will be applied to each of the three settings.

5.2. The E-Learning Setting

We have been applying the architectural framework to the E-Learning setting, which is a commercial Web-based English learning environment. In this setting, learners include a large number of male and female individuals with a variety of age. LCOPs include learning object developers, Web designers, and system managers.

Each component of the architectural framework in this setting contains information as follows:

1. *a learning history database* contains data for each learner obtained from his/her access log, such as answers, test scores, time taken to answer problems, the time taken to prepare, the number of repetitions, and the number of visits of descriptions.

2. *a learning object database* contains English learning materials, consisting of a set of learning themes, for instance, conversations at a store, or in an office.

3. *a learning object-part database* contains a set of tuples consisting of a problem, an answer to the problem, a description of the answer, relationships among the problems, and Q&A's posed by learners to the problem.

4. *user models* are generated using a feature vector consisting of a learning process (i.e., average scores, standard deviations, how scores changed over time, and how time spent on each problem changed over time), and a user profile (i.e., age, sex, academic background,

professions, marital status, and the duration of learning)

5. *community models* are generated by first applying collaborative filtering to the feature vector representing user models, and then by using a decision tree.

6. *feedback mechanisms* give learners feedback on their learning progress (e.g. score changes over time), and how and what learning materials their peer learners belonging to the same community model use.

By having this computational environment, learners can motivate themselves for learning by being able to compare their progress and performance with that of peer learners. They can also obtain information on how other learners use the E-Learning environment thereby enabling asynchronous, indirect collaboration. Learning object developers, on the other hand, can improve their learning materials by focusing on a particular group of learners by using the community models. Web designers can obtain ideas on how to improve the Web design by looking at how learners spend time in what part of which Web pages on the E-Learning system.

5.3. The Classroom Setting

We have applied the same framework for a junior level math class in a high school. In this setting, LCOPs are teachers, and learners are students.

1. *a learning history database* contains a set of each learner's problems attended, answers, scores, time taken to solve the problems, time taken to read the descriptions of the answers, time taken for preparation, and the number of repetitions.

2. *a learning object database* contains learning materials on mathematics, such as probability and sequences, according to the curricula designed by the teachers.

3. *a learning object-part database* contains a set of tuples consisting of a problem, an answer, description of the answer, the level of difficulty of the problem, and Q&A's about the problem.

4. *user models* use a feature vector consisting of time taken to prepare, the time of the day the students worked, the number of repetitions, average scores, standard deviations, correction rate, and how scores have changed over time.

5. *community models* are generated by first applying collaborative filtering to the feature vector representing user models, and then by using a decision tree.

6. *feedback mechanisms* give learners feedback on their learning progress (e.g. score changes over time and success rates), and how and what learning materials peer learners belonging to the same community model use.

By having this computational environment, students can motivate themselves for learning by being able to comparing their progress and performance with other classmates. They can also obtain information on how other students use the environment. Teachers can compare the feedback from the system with what the teachers have currently understood about each student through regular classroom teaching, and thus are able to better adapt to each student. Feedback is also helpful for teachers to redesign curricula, and improve teaching materials. Thus, the computational environment can be viewed as a way to broaden the communication channel between teachers and students.

5.4. The Open-Source Development Setting

Open-source software development can be viewed as community-based learning [15][25]. By sharing source codes and other software artifacts through the FTP site, the Web, mailing lists and E-mail, programmers gradually learn about the software.

In this setting, learners are end-users and new-comers to the community, usually peripheral participants of the open-source software community. LCOPs are core developers and members of the project.

1. *a learning history database* contains a Change Set and List maintained in a CVS system (see below) and the project Mailing Lists.

2. *a learning object database* is a CVS (Concurrent Versions System), which contain source code that project core members released on the project's Web server.

3. *a learning object-part database* does not explicitly exist but can be derived by decomposing software programs stored in *a learning object database* into software components.

4. *user models* contains a set of tuples consisting of:

   (a) the number of messages sent to the mailing lists,

   (b) the type of the problem dealt in the message (such as request, question, bug report, bug fix, and a patch program),

   (c) the type of contribution of the message (such as answers, questions, bug report, and bug fix), and

   (d) the referenced source code.

5. *community models* can be generated by using the above properties used in user models. Different from the other two settings, there may exist explicit community models in an open-source software development; for instance, core-members and end-users. We need to explore more on how those pre-existing community models can be taken into account in the generation of community models.

6. *feedback mechanisms* can help learners (end-users) by retrieving programs that are relevant to the user's characteristic usage. The LCOPs (core members) get feedback on the types and the number of communities who access to the source code.

The use of the architectural framework can help end-users and peripheral participants of the project to better find more relevant information from the open source resources. At the same time, core-members of the project can have better understanding of how the software artifacts are used by the community. Thus, the proper use of the architecture would help evolve the community development.

6 Acknowledgements

We would like to thank Jonathan Ostwald, Yunwen Ye, Yoshiyuki Nishinaka, Mitsuhiro Asada, Takeshi Fujioka and Tetsuo Ogino for their help in conducting this research. This research is partially supported by ACT-JST[1].

References

[1] A. W. T. Bates. *Technology, Open Learning and Distance Education*. Routledge, London, 1995.

[2] B. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13(6):4–16, 1984.

[3] L. Cuban. *Teachers and machines: The classroom use of technology since 1920*. Teachers College Press, New York, 1986.

[4] P. Ehn. *Work-Oriented Design of Computer Artifacts*. produced by Almquist & Wiksell International, 1988.

[5] G. Fischer, E. Arias, H. Eden, A. Gorman, and E. Scharff. Transcending the individual human mind– creating shared understanding through collaborative design. *ACM Transaction on Computer-Human Interaction (TOCHI)*, 7(1):84–113, 2000.

[6] D. Jonassen. *Computer in the Classroom: Mindtools for Critical Thinking*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1996.

[7] E. Kandogan. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 107–116, 2001.

[8] D. A. Kolb, R. Boyatzis, and C. Mainemelis. *Experiential learning theory: previous research and new directions*. Prepared for R. J. Sternberg and L. F. Zhang (Eds.), Perspectives on cognitive learning, and thinking styles, 2001.

[9] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and G. J. Riedl. Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–78, 1997.

[10] T. Koschmann. *Paradigm Shifts and Instructional Technology: An Introduction, In CSCL: Theory and Practice of an Emerging Paradigm*. Lawrence Erlbaum Associates, 1996.

[11] T. Koschmann. Logo-as-latin redux. *The Journal of the Learning Sciences*, 6:409–415, 1997.

[12] W. Lee and D. Owens. *Multimedia-Based Instructional Design*. Josey-Bass/Pfeiffer, San Francisco, CA., 2000.

[13] M. Morita. *Common Sense about E-Learning*. Asahi-Shinbunsha, 2002 (in Japanese).

[14] K. Nakakoji, Y. Yamamoto, and A. Aoki. Interaction design as a collective creative process. *Proceedings of Creativity and Cognition2002*, pages 103–110, October 2002.

[15] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye. Evolution patterns of open-source software systems and communities. *Proceedings of 5th International Workshop on Principles of Software Evolution (IWPSE2002)*, pages 76–85, May 2002.

[16] J. Ostwald. Knowledge construction in software development: The evolving artifact approach, 1996. Ph.D. Dissertation, Department of Computer Science, University of Colorado at Boulder.

[17] S. Papert. *Mindstorm: Children, Computers, and Powerful Ideas*. Basic Books, New York, 1980.

[18] D. Perkins. *Smart schools: From training memories to educating minds*. The Free Press, New York, 1992.

[19] J. Preece, Y. Rogers, H. Sharp, J. Wiley, and Sons. *Interaction Design: Beyond Human-Computer Interaction*. 2002.

[20] A. Repenning, A. Ioannidou, and J. Phillips. Collaborative use & design of interactive simulations. *Proceedings of Computer Supported Collaborative Learning Conference at Stanford (CSCL'99)*, pages 12–15, December 1999.

[21] M. Rosenberg. *e-Learning: Strategies for Delivering Knowledge in the Digital Age*. McGraw-Hill, 2001.

[22] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating.

[23] D. Smith, A.Cypher, and J.Spohrer. Kidsim: programming agents without a programming language. *Communications of the ACM*, 37:55–67, 1996.

[24] S. Waltzman. *Visual Deign Principles for Usable Interfaces, in The Human-Computer Interaction Handbook, J.A. Jack, A. Sears (Eds.)*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 2003.

[25] Y. Ye and K. Kishida. Toward an understanding of the motivation of open source software developers. *Proceedings of 2003 International Conference on Software Engineering (ICSE2003)*, 2003 (in print).

[1] "Research and Development for Applying Advanced Computational Science and Technology" of Japan Science and Technology Corporation