

Please STeP_IN: A Socio-Technical Platform for in situ Networking

Yoshiyuki Nishinaka¹
¹*SRA Key Technology
Laboratory
3-12 Yotsuya, Shinjuku
Tokyo, 160-0004, Japan*

nisinaka@sra.co.jp

Mitsuhiro Asada¹
²*Knowledge Interaction
Design Laboratory
RCAST, University of Tokyo
4-6-1 Komaba, Meguro
Tokyo 153-8904, Japan*

m-asada@sra.co.jp

Yasuhiro Yamamoto²
²*Knowledge Interaction
Design Laboratory
RCAST, University of Tokyo
4-6-1 Komaba, Meguro
Tokyo 153-8904, Japan*

yxy@comuter.org

Yunwen Ye^{1,3}
³*Department of
Computer Science
University of Colorado,
Boulder, CO80309, USA*

yunwen@colorado.edu

Abstract

This paper describes a system called STeP_IN (standing for Socio-Technical Platform for in situ Networking) that assists software developer to find and learn Java API libraries. It provides individualized search interface for Java developers, examples that illustrates the usage, and more distinctively a facilitating mechanism that connects Java developers to exchange expertise based on their expertise and social relations. STeP_IN is designed and developed based on the Dynamic Community theory, a new collaboration form that we have proposed.

1. Introduction

Object-oriented programming languages have become one of the widely used programming languages. A huge reusable class library is one of the major benefits brought by object-oriented programming languages. In fact, most of the productivity advantage of Java has come from the reuse of its accompanied API libraries [1]. While the huge and ever-growing library (see Figure 1 for the rapid growth of Java standard API library) has boosted the productivity of software development, it also poses great challenges for software developers to find, learn, and use those library components (classes and methods).

For software developers to use those library components successfully, they have to be able to

- (1) find appropriate components that are reusable in their current task;
- (2) understand the functionality of the components; and
- (3) integrate the components into their own development task.

This paper describes a system called STeP_IN (standing for Socio-Technical Platform for in situ Networking) that assists software developer to find and learn Java API libraries. It provides individualized search interface for Java developers, examples that illustrates the usage, and more distinctively a facilitating mechanism that connects Java developers to exchange expertise based on their expertise and social relations.

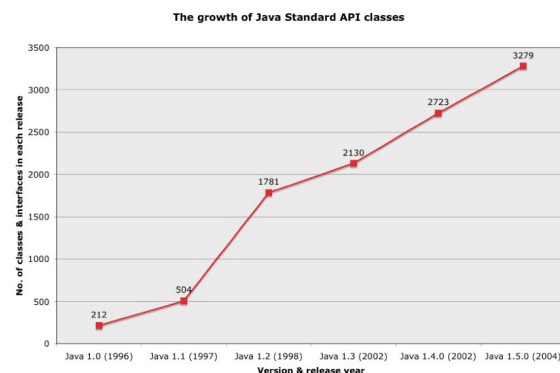
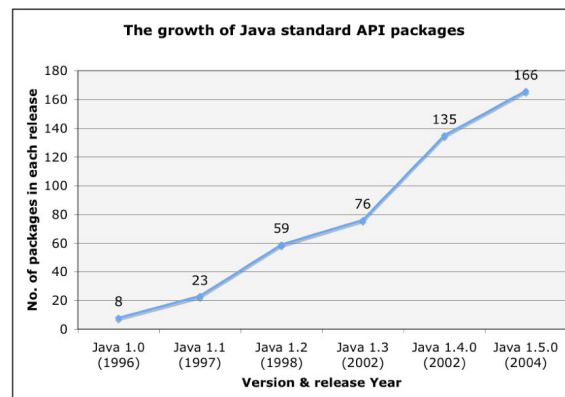


Figure 1: The growth of Java standard API

2. The problems with learning Java API

The Java API documentation system is the primary method for software developers to find and learn to use API library components. It features browsing structure that is organized based on packages and classes to assist software developers to find what they need; and its documents are meant to support software developers to learn about the functionality of each component. The support, however, is insufficient for two major reasons.

First, browsing is not scalable, and is difficult to use for software developers who are not quite familiar with the library and its structure. Users can easily get lost in the hyperspace provided by browsing mechanisms because in each browsing step, the user has to make the right choice, which is not an easy task. The needs for a reusable component come from a real development task, and in most cases, the developer only knows what functionality they are looking for, not necessarily knowing what packages and classes those components belong to. An efficient search interface therefore is needed.

Second, documentation only is not sufficient in many situations for software developers to understand how to use the component. Documentation is meant to explain the functionality of the component and describes the declarative knowledge of the component. For practical use, declarative knowledge alone is not enough because it is too abstract. Software developers need procedural knowledge too, and procedural knowledge is hard, if ever possible, to be explicitly stated in natural language. Procedural knowledge is better transferred through examples and through human interactions with experts.

Examples contextualize the abstract concepts of documents and explain to software developers the expected effect in an intuitive way. They provide useful aids for learning how to use, adapt, and combine the new information in their current task by drawing an analogy between the current task and the examples. Our previous empirical study has also found that whenever example codes exist, software developers always examine the example in order to understand the usage of library components [2].

Another effective way of learning to use a library component is to ask those who are experts on the given component. As one empirical study has found that [2] one important factor that differentiates expert developers from novices is that expert developers are able to “use other experts faster” to help them solve problems collaboratively because they often know the expertise of other experts and have a more reciprocal relationship with one another. The challenges here are:

- software developers may not know to whom they can turn for help; and
- experts who are able to help may not be willing to due to the interruption to their own work.

3. Overview of the STeP_IN system

The STeP_IN system that we are currently developing aims to provide a continuous support for software developers to learn Java library components. It not only provides sophisticated and personalized search interface for software developers to find the needed component based on functionality descriptions, but also facilitates social support for learning components from other experts.

The design of the overall system architecture is shown in Figure 2. The system consists of four major subsystems. At the core is the *database* subsystem, which stores the index of library components; the relationship between a developer and a component (technical profile); and the relationship between developers (social profile). The *profile management* subsystem is used by developers to initialize and update their social profile and technical profile. The *search engine* subsystem provides the personalized search interface and finds examples. The *dynamic community* (for details of this concept, see [3]; a short introduction will be given later) *support* subsystem identifies and chooses experts for developers to exchange information on a given component, and the discussions are archived in the database.

From a user’s perspective, the system works as follows:

- (1) A user (i.e. a Java developer) has to register to the STeP_IN server first.
- (2) After the user has registered, he or she need to download a profiling client program and use that to create and upload his or her initial technical profile and social profile. The technical profile represents what library components the user knows, and it is created by analyzing the Java programs the user has written. The social profile represents the user’s existing relationship with other STeP_IN users by analyzing his or her mailbox.
- (3) After the technical profile and the social profile are uploaded, the user can login to the system and start to search for library components.
- (4) The user types a natural language description of the functionality of the component that he or she is looking for.

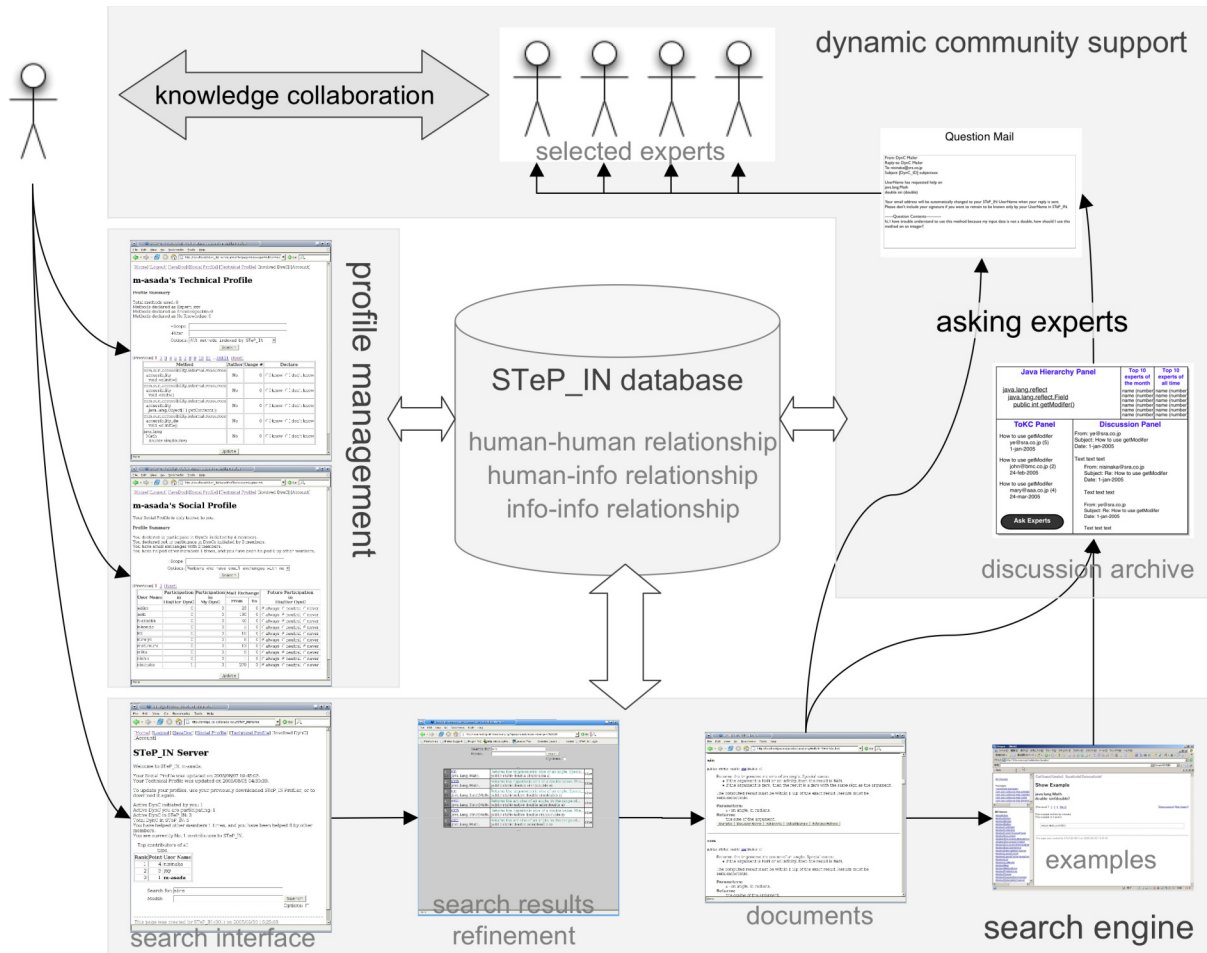


Figure 2: The system architecture of STeP_IN

- (5) Search results are returned. If too many search results are returned or the user does not find what he or she needs, he or she can use the search-by-refinement mechanism to refine the query or search range.
- (6) Clicking one of the returned results brings an enhanced Java document page. Each method in the Java document was enhanced with several new buttons that link to examples, discussion archive, and experts.
- (7) Clicking the *examples* link brings examples of the method if examples exist. In this page, users can upload their own examples too.
- (8) If the user clicks the *discussion archive* link (which appears both in the Java document page and the example page), archived discussions on this particular method are displayed.
- (9) If the user clicks the *ask experts* link (which appears in the Java document page, the example page, and the discussion archive page), he or

she will get an email composition window. The email will be sent by the system to selected members who are experts on the method and who are socially connected to the user who is asking for help.

- (10) The user, whom we call helpee, and the selected members, whom we call helper, can start knowledge collaboration on the method by exchanging emails, and the emails are archived in the database.

4. The search engine subsystem

STeP_IN treats each Java API method as an indexing and search unit. It uses the probability-based information retrieval technique, proposed by Robertson and Walker [4], to index and search library methods. User queries are written in natural language. Retrieval results are returned based on the computed similarity between user queries and the functionality description

STeP_IN supports personalized search. As mentioned in the previous section, users upload their technical profiles that represent what methods they have known by analyzing the Java programs that they have written. In STeP_IN, users can choose to limit their search range to all the methods that they have used, or to limit their search range to all the methods that they have never used. The former mechanism is meant to support the search of those methods that users vaguely know that they have used but could not remember the details; and the latter mechanism is meant to support the search of those completely new methods that users are looking for.

Java API libraries are separated into packages and classes. For most systems, only a small portion of the packages and classes are needed. If the search is limited to those packages or classes that are relevant to the current system development, search efficiency will be greatly improved. In the search result list returned by the search engine of STeP_IN, each method name is accompanied with the full class name that the method belongs to. As users move the mouse cursor over the package name, any subpackage name, or the class name, the name will be highlighted; and if users click the mouse, a small window will appear below the full class name (Figure 3). Users can click either the *+Scope* or the *-Filter* option.

all methods from the *java.lang.StrictMath* will not be returned as retrieval results.

1	sin	Returns the trigonometric sine of an
0.5	java.lang.Math.	public static double sin(double a)
2	sinh	Returns the hyperbolic sine of a doubl
0.5	java.lang.Math.	public static double sinh(double x)
3	sin	Returns the trigonometric sine of an a
0.5	java.lang.StrictMath. java.lang.StrictMath ±Scope ±Filter	public static native double sin(double
4	asin	Returns the arc sine of an angle, in the

Figure 3: Query-by-reformulation in context

5. The profile management subsystem


Step_IN : Profile Manager




Social Profile Technical Profile CLASSPATH: **D:\Projects\Dync\STEP_IN\lib\StPL71.jar**

package	class	method	# of definitions	# of references	check
java.lang	StringBuffer	java.lang.String	0	80	☒
java.lang	StringBuffer	java.lang.StringBuffer	0	56	☒
java.lang	StringBuffer	StringBuffer()	0	55	☒
pp.co.sra.smalitak	StObject	StObject()	1	40	☒
java.lang	Object	Object()	0	34	☒
java.awt	Point	Point(int, int)	0	29	☒
java.io	PrintStream	void	0	26	☒
pp.co.sra.smalitak	StImage	java.awt.Image.Buffer	1	27	☒
pp.co.sra.smalitak	StRectangle	StRectangle(int, int)	1	22	☒
java.lang	Object	java.lang.Class	0	22	☒
java.util	Vector	int.size()	0	21	☒
pp.co.sra.smalitak	StImage	int.width()	1	19	☒
pp.co.sra.smalitak	StImage	int.height()	1	18	☒
java.lang	StringBuffer	java.lang.StringBuffer	0	17	☒
pp.co.sra.smalitak	SmalltalkException	SmalltalkException	1	17	☒
pp.co.sra.smalitak	SmalltalkException	SmalltalkException	0	16	☒
java.util	ArrayList	int.size()	1	15	☒
java.lang	String	boolean	0	15	☒
pp.co.sra.smalitak	SmalltalkTestExa...	SmalltalkTestExa...	1	15	☒
pp.co.sra.smalitak	StObject	pp.co.sra.smalitak.St	1	14	☒
java.util	Vector	java.lang.Object	0	14	☒
pp.co.sra.smalitak	StBlockClosure	StBlockClosure()	1	14	☒
java.awt.event	WindowAdapter	WindowAdapter()	0	14	☒

Figure 4: Technical profile initialization

Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05)
0-7695-2465-6/05 \$20.00 © 2005 **IEEE**

Java programs and extract all the library methods that they have used (Figure 4). Users can uncheck the *check* field if they do not want a method to be included in their technical profile as a known method.

For social profile initialization, the user needs to specify a mailbox. The system will extract the mail addresses of those who have sent emails to the user and the number of the emails sent (Figure 5). Similarly, users can uncheck the item if they do not want to include in his or her social profile the mail exchange information with a particular person.

name	domain	# of messages	check
mailman-owner	lists.sra.co.jp	3	☒
nisinaka	sra.co.jp	2	☒
Yunwen.Ye	colorado.edu	19	☒
kumiyo	kid.rcast.u-tokyo.ac.jp	16	☒
k2	sra.co.jp	45	☒
yunwen	cs.colorado.edu	19	☒
matumoto	is.aist-nara.ac.jp	3	☒
matusta	ist.osaka-u.ac.jp	4	☒
ohira	empirical.jp	2	☒
matumura	sra.co.jp	6	☒
inoue	ist.osaka-u.ac.jp	6	☒
y23d	yahoo.com	4	☒
ysy	computer.org	4	☒
matumoto	is.naist.jp	11	☒
yzhong	hypersoft.com.cn	1	☒
ymtani	empirical.jp	1	☒
kaneda	ist.fujitsu.com	3	☒
masao	is.naist.jp	1	☒
yamada	kid.rcast.u-tokyo.ac.jp	2	☒
yunwen	colorado.edu	2	☒
yaroslav	scam.xcf.berkeley.edu	1	☒
yosh	gimp.org	145	☒
che	debian.org	13	☒

Figure 5: Social profile initialization

Users can also update their profiles through the STeP_IN server at any time. As shown in Figure 6, the user can explicitly indicate whether he or she knows the method no matter he or she has used it or not.

Figure 7 shows the interface for updating social profile. Initial social profile only shows the number of emails that the users have exchanged with other members, and a complete social profile in STeP_IN includes also the interaction history between the user and other members in STeP_IN. The second column *Participation in his/her DynC* shows how many times the user has tried to help the member by participating in a dynamic community (shorthand as DynC) created for that member. The third column *Participation in My DynC* shows how many times the member has tried to help the user by participating in the DynC created for the user. Both numbers as well as the number of email exchange are facts that cannot be changed. However, the user can specify whether he or she is willing to continue knowledge collaboration with the member in the future by choosing one of the three options in the last column *Future participation in his/her DynC*.

Method	Author	Usage	Declare
com.sun.accessibility.internal.resources.accessibility void <init>()	No	0	☐ I know ☐ I don't know
com.sun.accessibility.internal.resources.accessibility void <init>()	No	0	☐ I know ☐ I don't know
com.sun.accessibility.internal.resources.accessibility java.lang.Object[] getContents()	No	0	☐ I know ☐ I don't know
com.sun.accessibility.internal.resources.accessibility de void <init>()	No	0	☐ I know ☐ I don't know
java.lang.Math double sin(double)	No	0	☐ I know ☐ I don't know

Figure 6: Update technical profile

User Name	Participation in His/Her DynC	Participation in My DynC	Mail Exchange From	To	Future Participation in His/Her DynC
akiko	0	0	23	0	☐ always ☐ neutral ☐ never
aoki	0	0	130	0	☐ always ☐ neutral ☐ never
h-asaka	0	0	40	0	☐ always ☐ neutral ☐ never
k-kondo	0	0	5	0	☐ always ☐ neutral ☐ never
k2	0	0	10	0	☐ always ☐ neutral ☐ never
kumiyo	0	0	9	0	☐ always ☐ neutral ☐ never
matumura	0	0	13	0	☐ always ☐ neutral ☐ never
mika	0	0	9	0	☐ always ☐ neutral ☐ never
nishis	0	0	1	0	☐ always ☐ neutral ☐ never
nisinaka	1	3	278	3	☐ always ☐ neutral ☐ never

Figure 7: Update social profile

6. The dynamic community support subsystem

When reading the document is not enough for software developers to understand how to use the library method, they need further help from other expert members. The dynamic community support

subsystem is meant to provide the social support for understanding library method of interest by creating a temporal social platform for in situ knowledge collaboration.

The dynamic community support is activated when users click the *Ask experts* link and send out a question email requesting for help on a particular method. It selects a small group of users for the request for help email to be sent to, provides communication support for subsequent knowledge exchanges between the selected group members (helper) and the user (helpee), and archives the discussion that takes place.

The small group of experts is chosen based on the principles of the new conceptual framework Dynamic Community that we have proposed to support situated knowledge collaboration (see the next section for a brief introduction, for more details see [4]). When the selected experts exchange messages with the developer who is seeking for help, those messages are stored in the discussion archive associated with the method, so that other developers who have similar problems later can benefit by clicking the *Discussion archive* link.

6.1 Dynamic Community

A dynamic community, or DynC, is dynamically formed in a knowledge work space. A knowledge work place consists of a group of knowledge workers, and the knowledge that the workers hold. The group can either be a formal organization such as a company or an informal online community. A DynC is a subgroup of knowledge workers that forms ad hoc in support of a particular user and a particular task, and dissembles as the task is finished. Unlike a static community that forms around a particular domain and exists for a long time, a dynamic community forms for a particular task, and exists only for a short period. The members of a DynC are selected using the following two criteria:

- (1) They have expertise on the particular task, and
- (2) They already have social contacts with the particular user.

The first criterion is grounded in the observation that in today's highly specialized world expertise is no more an absolute attribute of person but a relative function of a person and a task. In other words, omniscient experts for many domains do not exist any more, and experts can be identified only after the task is known. The second criterion is grounded in the assumption that existing social contacts between the helpers and the helpee could provide extra motivation to engagement in knowledge collaboration based on the social norm of generalized reciprocity [5].

In summary, a DynC is task-specific: for different tasks, different DynCs will form even for the same user because different groups of experts will be identified; a

DynC is user-specific: for different users, different DynCs will form even for the same task because users has different social contacts; and a DynC is in-situ and ephemeral: it is created dynamically when a need for collaboration arose in a real task and dissembles when the task is solved.

6.2 Creating Dynamic Communities in STeP_IN

The STeP_IN system maintains, in its central server, a database that stores the technical profile and social profile of each user. Both profiles are used to create a dynamic community for a particular user and a particular method.

When a user (i.e. a helpee) clicks the *Ask experts* link, according to the principles of DynC, STeP_IN goes through two steps:

- (1) expert identification, and
- (2) expert selection

to create the group of experts who should receive the help request.

6.2.1 Expert identification. The expert identification process examines the relation from the method to the experts in the database to find all the members who probably know the method, and create an ordered list of *Candidate Helpers*. *Candidate Helpers* are identified in the following order based on three levels of expertise.

- **Confirmed expertise:** The member has participated in a previous discussion (sent a message to a previous DynC on the same method) and the original helpee ends the DynC with a satisfactory evaluation. If multiple members have confirmed expertise, they are listed in the order of the number of helpful DynCs on the same method that they have participated.
- **Claimed expertise:** The member has checked the *I know* field in his/her technical profile (Figure 6).
- **Inferred expertise:** The member has usage counts in his/her technical profile, and did not check the *I don't know* field, and members are listed in the order of the number of usage count.

6.2.2 Expert selection. The expert selection process examines the social relation from the helpee to other members in the database to select a small group of people from the list of *Candidate Helpers*.

Social relations are computed from each user's social profile. Before describing the process of expert selection, we first define the following social relations.

- *help* $\langle A, B, t \rangle$: *A* helped *B* at time *t*. This means that *A* sent at least one message to the DynC

initiated by B at time t over some method. To note this is quite different from the definition of *Confirmed expertise*. In *Confirmed expertise*, the DynC has to be on the same method, and the DynC has to be evaluated by B as useful. In this relation, A could have participated in any DynC over any method, regardless the DynC was helpful or not.

- $friend\langle A, B \rangle$: A claims that A is a friend of B . This means that A has chosen, in A 's social profile, the *always* field under B 's name. For example, in Figure 7, $friend\langle m-asada, akiko \rangle$ exists. This relationship can only be seen by A for the protection of privacy. It is also directional, namely, $friend\langle A, B \rangle$ does not mean $friend\langle B, A \rangle$.
- $exclude\langle A, B \rangle$: A states that he or she does not want to engage in collaboration with B , namely, A has chosen, in A 's social profile the *never* field under B 's name. Similar to *friend*, this relationship is directional and can only be seen by A (B would never know this).
- $email\langle A, B \rangle$: The number of emails that A has sent to B .

With the relations defined above, STeP_IN uses the following selection rule to choose from the ordered list of *Candidate Helpers* those people who have the most expertise and yet most likely to help person A on the particular method. The goal of this selection process is to choose a predefined number (for instance, 5) of experts to participate in the DynC for A . There are six steps in the selection rule, and when the predefined number of experts has been found at any step, the remaining steps are ignored.

Selection Rules: For each person X in the *Candidate Helpers* list.

1. If $exclude\langle X, A \rangle$ exists, X is removed because X has indicated no intent to collaborate with A .
2. If $friend\langle X, A \rangle$ exists, X is selected because X claims to be a friend of A , so X would probably help A .
3. If $|help\langle A, X, t \rangle|$ is greater than $|help\langle X, A, t \rangle|$, X is selected because A has helped X more times than X has helped A , X should help A .
4. If $help\langle A, X, t \rangle$ and t is recent than $help\langle X, A, t \rangle$, X is selected because A has just helped X on other information recently.
5. In the remaining list of *Candidate Helpers*, choose the persons who have sent most emails to A , based on $email\langle X, A \rangle$, because X must know A quite well and might be willing to help A if he or she has sent many emails to A .
6. In the remaining list of *Candidate Helpers*, choose the persons who have been helped most in the whole STeP_IN regardless who have helped them,

because although they do not have direct social relationship with A , they should offer help due to the principle of generalized reciprocity in the whole group. For those persons who have the same number of being helped, choose those who have been helped by someone most recently.

The selection process gives the priority to personal preferences (the relationships of *exclude* and *friend*) and uses less reliable social relationship calculated from email exchange history as the backup mechanism for selection. However, email exchange history will be heavily explored in the initial deployment of STeP_IN.

6.3 Supporting knowledge collaboration

The request for help email is then sent to the selected DynC members. A DynC member who wants to help can send a reply to the same group, and other DynC members can comment on each other's message and engage in knowledge collaboration on the method that they are interested in and have certain expertise. When the helpee feels the discussion is sufficient, he or she can close the DynC by going to the STeP_IN server and gives his evaluation of the DynC as helpful or not. The "helpful" evaluation will change the expertise of all the DynC members on the method to *confirmed expertise*. All the discussions are archived and linked to the method for the benefit of later users who have similar questions on the same method to avoid repeated discussions of the same kind that wastes the time and talent of experts.

Special care has been taken in the STeP_IN system to avoid forcing experts into collaboration because we believe for sustaining the long-term success of knowledge collaboration, experts have to be able to participate in DynC out of their true free will of sharing instead of giving in to potential social pressures.

When a helpee requests help from experts he or she knows well, the social pressure is often shift on the experts because they might fear potential social backlash if they do not provide help and violate the rule of reciprocity. In such cases, the experts are often forced into collaboration even if they might not in a condition or a mood for collaboration at this particular moment. This would be detrimental to the long-term sustainability of the collaboration platform and some experts might simply withdraw from it. We believe avoiding such forced collaboration is essential for creating a sustainable voluntary collaboration platform that relies on the good will of benefactors.

Therefore, in STeP_IN, when the request for help is sent out to the DynC members, the helpee does not know to whom the request is sent, so that even if a

DynC member does not offer help due to various reasons, the helpee would not notice. However, any help message from a DynC member to the DynC shows its origin so that the DynC member's contribution to the group is publicly recognized and properly acknowledged.

Another strategy to respect the will and time of experts is that experts can withdraw from participating in any DynC at any time, or refuse to collaborate with any particular person.

Accompanying the request for help email, a notification email is also sent to the DynC members to notify that they have been asked to help user *A* on method *M*. In this notification email, three links are provided for DynC members to avoid forced collaboration. The first link allows the expert to withdraw quietly from the current DynC if he or she is not interested at the moment, so that he or she would not receive further emails of this DynC. The second link allows the expert to refuse to receive any further request for help from *A* by change his social profile to *never* under *A*. The third link allows the expert to stop receiving any further request for help on method *M* by changing his technical profile field of *M* to *I don't know*. All three actions are only known to the expert himself and no other people (neither *A* nor other DynC members) would notice.

7. SUMMARY

This paper describes the STeP_IN system that we are currently developing to support Java developers to collaboratively learning the vast amount of Java API library that contributes to the rapidly increased productivity of software development. STeP_IN provides continuous support for learning by offering individualized search interface, adopting user modeling techniques to identifying experts for help, and utilizing social networks to find experts who are most likely willing to help a particular user. The system is still under development and the whole interface needs to be polished for real use. Our future work includes continued implementation and user study to verify the usefulness of the system.

ACKNOWLEDGMENTS: This research is supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan, Grant 15103 of the Open Competition for the Development of Innovative Technology program.

References

- [1] P. van der Linden, Just Java 2, 4th ed. Upper Saddle River, NJ: Prentice Hall, 1998.
- [2] Y. Ye and G. Fischer, "Information Delivery in Support of Learning Reusable Software Components on Demand," presented at Proceedings of 2002 International Conference on Intelligent User Interfaces (IUI'02), San Francisco, CA, 2002.
- [3] Y. Ye, Y. Yamamoto, and K. Kishida, "Dynamic Community: A New Conceptual Framework for Supporting Knowledge Collaboration in Software Development," presented at Proceedings of 11th Asia-Pacific Software Engineering Conference (APSEC 2004), Busan, Korea, 2004.
- [4] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," in The 3rd Text REtrieval Conference (TREC-3), D. K. Harman, Ed. Gaithersburg, MD: National Institute of Standards and Technology, 1995, pp. 109-126.
- [5] A. Mili, S. Yacoub, E. Addy, and M. Hafedh, "Toward an Engineering Discipline of Software Reuse," IEEE Software, vol. 16, pp. 22-31, 1999.
- [6] M. D. Williams, "What Makes RABBIT Run?" in International Journal of Man-Machine Studies, vol. 21, 1984, pp. 333-352.