Presented at the CHASE (Human and Cooperative Aspects of Software Engineering) Workshop at ICSE2008, Leipzig, Germany, May, 2008.

The Economy of Collective Attention for Situated Knowledge Collaboration in Software Development

Yunwen Ye[1,2]
[1]Dept. of Computer Science
University of Colorado
Boulder, CO80309, USA
yunwen@colorado.edu

Kumiyo Nakakoji[2,3]
[2]SRA Key Technology Lab
3-12 Yotsuya, Shinjuku
Tokyo, 160-0004, Japan
kumiyo@kid.rcast.u-tokyo.ac.jp

Yasuhiro Yamamoto[3]
[3]KID Laboratory, RCAST
U. of Tokyo, 4-6-1 Komaba, Meguro,
Tokyo, 153, Japan
yxy@kid.rcast.u-tokyo.ac.jp

ABSTRACT
Because the knowledge required for the construction of a complex software system is often widely distributed among its members, programmers routinely engage in collaboration with each other to acquire knowledge resided in the heads of their peers to accomplish their own programming tasks. We call this kind of collaboration *situated knowledge collaboration*. Situated knowledge collaboration comes with costs and the costs vary depending on the communication mechanism used. To better understand the cost-benefit structure of different communication mechanisms in support of situated knowledge collaboration, we propose the conceptual framework of collective attention economy. The analytic power of the conceptual framework is illustrated in the comparison of two communication mechanisms.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *computer-aided software engineering.* H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces - *computer-supported cooperative work, theory and models.*

General Terms
Design, Theory.

Keywords
Collective attention economy, situated knowledge collaboration

1. INTRODUCTION
Programming is essentially a knowledge construction process during which programmers apply a wide variety of knowledge from the computer and application domains to construct new knowledge artifacts—software. As software systems become increasingly complex and large, and the development teams become increasingly distributed, communication and collaboration become the more important factors that determine the productivity and quality of software development. The needs for communication and collaboration arise from two major areas. The first one is the consequences of division of labor: a complex software system needs to be built by many hands. The second one is the consequences of distribution of knowledge: the required knowledge to build a software system becomes so vast that no single programmer can have all the knowledge, and software development therefore requires the integration of knowledge distributed in many heads.

Despite decades of software engineering research that tries to decompose software development into independent and parallelizable tasks, programming tasks are still helplessly interdependent in a complex way [4]. The decomposition of tasks also deepens the distribution of knowledge: each programmer only has partial knowledge of the system and of the process. Due to the inter-dependency, each programmer often needs to seek knowledge from peers to carry out his/her work efficiently and effectively. We call this kind of collaboration *situated knowledge collaboration* in which a programmer asks others for expertise necessary to solve his/her immediate programming task at hand.

Situated knowledge collaboration comes with costs. The costs could even outweigh its benefits and lower the group productivity [12]. Such costs vary depending on what communication mechanisms are used. To understand the cost better, we introduce the notion of *collective attention* to represent the total cost of attention of all parties involved in an act of situated knowledge collaboration, and propose a conceptual framework called *economy of collective attention* as an instrument to analyze the cost-benefit structures of various communication mechanisms that are used to support situated knowledge collaboration in software development.

2. SITUATED KNOWLEDGE COLLABORATION IN PROGRAMMING
Many types of needs for communication and collaboration exist in software development, such as the needs of informing programmers of the status of the project, the needs of brainstorming for design ideas, and the needs for consensus building. Situated knowledge collaboration is different from other needs for communication and collaboration, and has the following distinctive characteristics:

- It arises on an as-needed rather than scheduled basis, and arises from the working context of an individual programmer.
- It is not for the general purpose of learning or creating awareness in which information is not immediately coupled with the task at hand. Rather it serves for solving an immediate problem of the programmer, and requires quick resolution.
- It occurs as sequences of highly focused interactions in a short period of time, with a relatively small group of participants.
- It is mainly for the benefit of the expertise-seeking programmer, but has direct impact on the productivity of the whole group because all programmers involved in situated knowledge collaboration also belong to the same group.

- It recurs frequently and a programmer often assumes the role of expertise-seeker or expertise-provider at different times.

The above settings of situated knowledge collaboration entail the following three constraining factors.

(1) *The collective attention has a limited capacity due to the fixed size of the project group.* This is different from volunteer-based community projects where the number of "eyeballs" can be increased through strategies of turning passive users into active contributors.

(2) *Situated knowledge collaboration is not a one-time affair; it has to be sustainable* because its continuous enactment is required throughout the lifecycle of the project consisted of relatively stable members. Engagement in one collaborative act should not result in one's reluctance to participate in further collaboration acts down the road.

(3) *The costs and benefits of situated knowledge collaboration have to be considered together with the group productivity* that is of essential importance. If a programmer is unable to obtain such information in the head of peer programs timely, he or she cannot carry out his or her programming effectively or efficiently, and thus lowers his or her own productivity, which in turn lowers the productivity of the project team. On the other hand, if a programmer is frequently interrupted by being asked to provide help, then his/her productivity is significantly reduced, resulting in lower group productivity.

3. COLLECTIVE ATTENTION ECONOMY

Attention is an intrinsically scarce resource because everyone has only a certain stock of supply. We are entering a world where our lives are guided more by the laws of the economics of attention because attention is quickly becoming the scarcest resource in our society [7]. *Attention economy* is concerned with the patterns of allocating attention for the best possible benefits. Allocating attention to a person is to turn one's mind to align with others', and allocating attention to a thing is to act or reflect upon it. Allocating collective attention in situated knowledge collaboration is to align helpers' minds with those of askers. From the perspective of *economy of collective attention*, however, the allocation of collective attention should not only match the interests of the asker, but also match the interests of the helpers and, more importantly, the productivity of the group.

3.1 The Cost of Collective Attention

In an act of knowledge collaboration, both the asker and potential helpers consume attention in communication with each other.

An asker needs to find where the needed expertise is located, and who potentially has the expertise. Previous research has shown that such transactive knowledge, defined as an awareness of who knows what, takes extensive time to develop, and its utilization consumes intensive attention [11]. We denote the attention cost for finding the location of expertise as C_{Find}.

The question needs to be formulated and articulated, and we denote the attention cost for this act as C_{Ask}. The way that the question is presented affects the response it will receive [1]. The asker also needs to make a decision based on social cues whether the helper could be interrupted [8], and to determine opportune times to interrupt [3].

When potential helpers are presented with a question, all of them are interrupted and distracted from their current work. The cost of attention (denoted as $C_{Interrupt}$) includes not only the attention spent on attending to the interrupting event but the disruption of flow and the accompanied work resumption efforts [14].

Some of those who are presented with a question have to make a conscientious decision to respond to or ignore it. A number of factors are brought into consideration in this decision-making process: whether they have sufficient expertise on the topic [15]; how many efforts does it take to post a reply [10]; how they perceive their relationship with the asker [2]; and their eagerness of offering help. To make this decision, they at least need to skim the question by finding out the asker and the topic [9]. We denote the attention cost for this process as C_{Skim}.

If a helper decides to respond to the question, s/he needs to spend time and attention in thinking and composing the response. The cost of attention for answering the question is denoted as C_{Answer}.

Upon receiving an answer, the asker needs to evaluate its quality and interpret its meaning in terms of his/her task. Not all responses are of equal value and quality. The perceived expertise of the helper and previous favorable interactions with him/her assist askers in evaluating the answer [5]. We denote this cost of attention as $C_{Evaluate}$.

Assume a question is sent to N potential helpers. The *Cost of Collective Attention* (CoCA) consumed for the communication can be modeled as follows:

$$\begin{aligned} CoCA = {} & C_{Find} + C_{Ask} && \text{// the cost of the asker} \\ & + N*C_{Interrupt} + p*N*C_{Skim} && \text{// the cost of potential helpers} \\ & + q*N*C_{Answer} && \text{// the cost of helpers} \\ & + C_{Evaluate} && \text{// the cost of the asker} \end{aligned}$$

where p is the ratio of potential helpers who skim question for deciding whether to reply ($0 \le p \le 1$), and q is the ratio of helpers who actually reply ($0 \le q \le p$).

3.2 Benefits

The benefits of situated knowledge collaboration come from the asker, the helper, and onlookers. The asker obviously obtain the most significant benefits by either saving his/her own time of finding the necessary knowledge or gaining new knowledge that s/he has not had. The benefit for the helper is mostly indirect. The helper gains social capital that includes social recognition and easiness of obtain help down the road.

Members who are passively involved in situated knowledge collaboration can have two benefits: an increased awareness of what the whole group is up to, and a serendipitous learning opportunity of acquiring new knowledge. Benefits of onlookers do not need to be obtained at the same time when the situated knowledge collaboration takes place. If the collaboration can be archived and become accessible, other people can still obtain the "onlooker benefits" without being a real-time onlooker.

4. Cost Comparison Study through Simulation

To illustrate the analytic power of the collective attention economy, we conducted a simulation study to compare the cost-benefit structure of two communication mechanisms: conventional mailing list (ML) and the *ephemeral mailing list* (EML) that we have implemented in the STeP_IN system [16].

The EML mechanism works as follows. Whenever a programmer asks for information from his peers, s/he posts a question in the same way as to a mailing list, without a prior knowledge of who

the experts are. In other words its C_{Find} is as low as that of ML. EML differs from ML in that not all mailing list members will receive the question. Instead, it routes the question to a very small subset of the whole group that are automatically chosen based on their *technical expertise* on the topic of the question and *social relationships* with the asker. In other words, only those who are able to and most likely willing to provide answers are interrupted with the question, and the majority of other members who are neither interested in the topic nor interested in helping the asker do not need consume their attention on this particular question. The question and answers are then archived in a repository to retain the onlooker benefits for those members who do not directly participate in the situated knowledge collaboration. Details of the mechanism are described in [16].

4.1 Simulation Setup

The simulation study uses the mailing list archive of *Lucene-Java* project (an open source project) in the period between 2001 and 2006. We have simulated how situated knowledge collaboration would have taken place if the project had been using the EML instead of conventional ML.

The 17,942 messages sent between 2001 and 2005 are used as the base data to set up the *technical expertise* profiles and *social relationship* profiles of the 2,282 members of Lucene-Java ML. We then simulated how EMLs would be generated for the questions posted in 2006, and compared the results with the actual conversation threads in the mailing list. As shown in Table 1, among the 20 threads that we simulated the generation of EMLs in the STeP_IN system, seven cases show that all the actual repliers were included in the simulated EMLs; eight cases show some actual repliers were included; and five cases have no matching. This means that if the Lucene-Java project had used the EML mechanism, 15 askers (75%) would have been able to get responses from their peers.

Table 1: Simulation results

Thread No.	Subject	Asker	Actual repliers in the Lucene ML				EML members selected in STeP_IN				
1	*My first question in 2006 :D*	U1293	U1156				U0126	U0256	U0065	U1300	U1186
2	*numDocs() after undeleteAll()*	U1671	U1162				U0568	U1102	U1162	U1978	U1086
3	*Lock obtain timed out + IndexSearcher*	U1605	U0065	U1162			U0126	U0065	U0703	U0561	U1185
4	*Generating phrase queries from term queries*	U0390	U1162	U0953	U0549	U1286	U0292	U1558	U0953	U1787	U0775
5	*How do I get a count of all search results inside of my content?*	U1858	U1286				U0953	U0292	U0126	U1799	U1286
6	*What's the differents between QueryParser and Query*	U1856	U0126				U1034	U1293	U1329	U1307	U0070
7	*Range queries*	U1871	U0126	U1291	U1286	U1754	U1117	U0321	U1070	U1162	U0126
8	*How does the lucene normalize the score?*	U1214	U1276	U1823	U1286	U1162	U1823	U1276			
9	*Sorting by Score*	U1757	U0499	U1286			U0126	U1286	U1034	U1184	U0776
10	*index merging*	U1409	U1162	U1246	U1772		U1246	U1949	U2161	U1845	U1622
11	*How to get mapping of query terms to number of their occurrences in a doc?*	U1837	U1286	U0126	U0065		U1817	U0720	U1286	U1162	U1795
12	*query formulation*	U1606	U1162				U2061	U1162	U1286	U1777	
13	*How can I get a term's frequency?*	U1928	U1772	U0849			U1198	U1144	U1356	U0628	U0553
14	*Efficiently updating indexed documents*	U1926	U1162				U1952	U1162	U1074	U1456	U1978
15	*Help on Similarity*	U1777	U1286				U1499	U0789	U0873	U0325	U0292
16	*sumOfSquaredWeights for lengthNorm*	U1777	U1286				U0126	U1286	U1192	U1149	U1230
17	*Get only count*	U1792	U0390	U1162	U0953		U0531	U0100	U1192	U1684	U1149
18	*Reading stop word from a file!*	U1778	U1978				U1978				
19	*Changing ranking*	U1884	U0065	U1582	U1286		U2055	U1286	U1963	U1162	U2108
20	*writeChars method in IndexOutput*	U2009	U1162				U1162				

4.2 Exploration

We now try to estimate how the EML mechanism changes the cost structure of collective attention economy. We estimate the cost of attention by multiplying the number of words in each message and reading rate. We use 550 words per minute as the skimming rate (http://en.wikipedia.org/wiki/Reading_rate), and 300 seconds as the time to compose a message according to the data reported in [10].

For the sake of simplicity, we assume that 2,282 users who posted messages are all members of the Lucene-Java ML. From our own experience, we think it is reasonable to assume that not everybody in the mailing list reads all messages. We therefore varied the value *p* of the CoCA formula as follows: the number of users who skim the first message of a thread, and the number of users who skim the entire thread. Five sets of values were used: (a) 100% skim the first message and 10% skim the entire thread; (b) 50% skim the first message and 10% skim the entire thread; (c) 20% skim the first message and 5% skim the entire thread; (d) 10% skim the first message and 1% skim the entire thread; and (e) 10% skim the first request and only repliers read entire thread.

Figure 2 shows the collective cost of attention spent per thread by using the mean number of word counts for the whole mailing list. If 100% of the Lucene-Java ML members skim the first message of a thread, it would consume 1,001.80 minutes, which is more than 16 hours. If 10% skim the first message, which is 228 members, it takes 80 minutes in total. Lucene-Java had 5,693 threads in total and 1,121 in 2006, so the total cost of consumed collective attention is quite large, which is 2,376 weeks in total and 506 weeks in 2006 if all the 2,282 users at least skimmed the initial message of each thread (Figure 3).

As shown in Figures 2 and 3, using EML would significantly decrease the cost of collective attention, as the number of users who skim the messages but do not reply is greatly reduced. For the 5 sets of *p* value, the reduction rates are 98%, 97%, 94%, 85%, and 80% respectively.

This reduction could result in the lost opportunities of obtaining help. For example, in five cases (Table 1), the repliers were not selected to EML, and in the other 8 cases, some users who were not selected to EML also contributed their expertise. If we consider only the five cases as failures of knowledge collaboration, then the benefits of EML drops by 25%; if we consider both five cases and eight cases as failures (which is very strict), the benefits of SIJ drops by 65%.

In fact, in the Lucene-Java mailing list, 20% questions (1,228 out of 5,693) did not receive a single reply. Even if we assume only 10% of the ML members skims this kind of message, it still takes 1,541 hours that are completely wasted. If the same 1,228 questions were posted in EML and did not receive an answer, the wasted cost is 291 hours.

4.3 Discussion

This simulation experiment only shows how EML has a different cost-benefit structure as mailing lists. The variables used here are not accurate; they are based on educated guess that draws from existing studies. Those variables are likely to change in different situations. However, this illustrates that CoCA formula we proposed can be used to estimate the collective attention cost of situated knowledge collaboration with different communication mechanisms, and such estimation could provide the basis for programming teams to make informed choice about the right channel for their different collaboration needs.

The benefits difference of the two communication channels is mainly dependent on the accuracy of identifying the right experts, which in turns depends on the precision of the profiles of technical expertise and social relationships. As Table 1 shows, profiles that are mined from discussions in the mailing list only can have 25%-75% accuracy in identifying the right group of experts. We are cautiously optimistic that this success rate could improve further if we are able to create technical profiles by also analyzing the programs that each member has created.

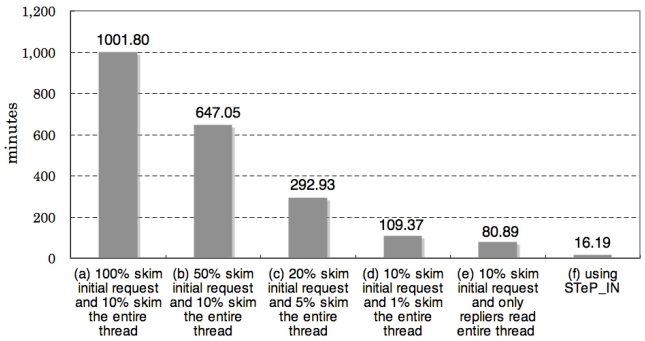

Figure 2: Collective Cost of Attention Per Thread

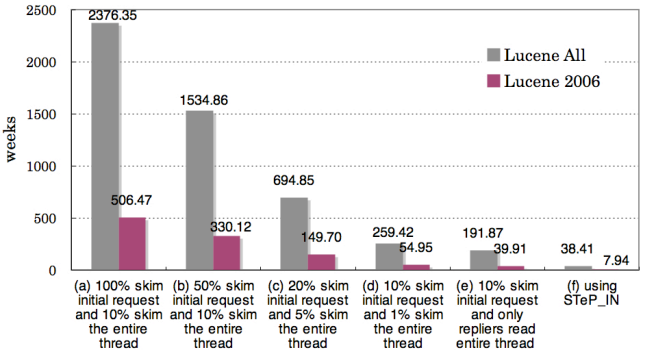

Figure 3: Collective Cost of Attention in Total

5. Concluding Remarks

The needs to balance attention and communication are recognized in [6], which suggests two strategies to conserve attention resources in communication by providing information asynchronously and by reducing the frequency of interruption through the aggregation of information. These strategies can be subsumed in reducing the cost of $C_{Interrupt}$. However, as we can see from the formula, this cost is only a portion of the cost of collective attention in collaboration.

We have seen many studies that point out the cost of interruption and the overload of communication brought by ubiquitous connectivity [3], but we still do not have a systematic way to understand how to address the "dearth of attention" resulted from those technologies. We are fully aware that to model concepts as complicated and subjective as attention and collaboration should not be taken lightly. The proposed formula is not meant to capture everything. The main goal is to use this relatively simple framework to analyze the factors that affect the economic utilization of the collective attention of all parties involved, either actively or passively, in situated knowledge collaboration. The conceptual framework is not meant to cover all types of collaboration that take place in software development but focusing on situated knowledge collaboration only. A study has shown that this type of ad hoc and situated knowledge collaboration takes up to 41% of the programmer's time [13], therefore improving the economy of collective attention for this type of collaboration has a major impact.

This paper has shown how to use the notion and formula of the cost of collection attention to compare different communication mechanisms and how it can help design new communication mechanisms that have different cost-benefit structure by manipulate some of the variables. By trying to change the number *N*, we came up with the EML mechanism that is neither direct email nor mailing list, but something in between email and mailing list with the feature of persistent storage of discussions. The comparison is not meant to rank the absolute superiority of communication mechanisms, but give a clear understanding of each mechanism so that programming teams can choose the most appropriate communication channel for their varied knowledge collaboration needs in their specific socio-technical environment.

6. References

[1] Arguello, J., et al., Talk to Me: Foundations for Successful Individual-Group Interactions in Online Communities, in *Proceedings of CHI06*. 2006, p. 959-968.

[2] Cross, R. and S.P. Borgatti, The Ties That Share: Relational Characteristics That Facilitate Information Seeking, in *Social Capital and Information Technology*, M. Huysman and V. Wulf, Editors. 2004, The MIT Press: Cambridge, MA. p. 137-161.

[3] Dabbish, L.A. and R. Kraut. Controlling Interruptions: Awareness Displays and Social Motivation for Coordination, in *Proceedings of CSCW2004*. 2004.

[4] de Souza, C.R.B., et al., How a Good Software Practice Thwarts Collaboration: The Multiple Roles of APIs in Software Development, in *Proceedings of FSE04*. 2004: Newport Beach, CA. p. 221-220.

[5] Flammer, A., Towards a Theory of Question Asking. *Psychological Research*, 1981. **43**: p. 407-420.

[6] Fussell, S.R., et al., Coordination, Overload and Team Performance: Effects of Team Communication Strategies, in *Proceedings of CSCW98*. 1998: Seattle WA. p. 275-284.

[7] Goldhaber, M.H., The Attention Economy. *First Monday*, 1997. **2**(4).

[8] Herbsleb, J.D. and R.E. Grinter, Architectures, Coordination, and Distance: Conway's Law and Beyond. *IEEE Software*, 1999. **1999**(September/October): p. 63-70.

[9] Jackson, T., R. Dawson, and D. Wilson, The Cost of Email Interruption. *Journal of Systems and Information Technology*, 2001. **5**(1): p. 81-92.

[10] Lakhani, K.R. and E. von Hippel, How Open Source Software Works: Free User to User Assistance. *Research Policy*, 2003. **32**(6): p. 923-943.

[11] McDonald, D.W. and M.S. Ackerman, Just Talk to Me: A Field Study of Expertise Location, in *Proceedings of CSCW'98*. 1998: Seattle, WA. p. 315-324.

[12] Reder, S., The Communication Economy of the Workgroup: Multi-Channel Genres of Communication, in *Proceedings of CSCW1988*. 1988, ACM Press: New York. p. 354-368.

[13] Robillard, P.N., The Role of Knowledge in Software Development. *CACM*, 1999. **42**(1): p. 87-92.

[14] Szoestek, A.M. and P. Markopoulos, Factors Defining Face-To-Face Interruptions in the Office Environment, in *Proceedings of CHI06*. 2006. p. 1379-1384.

[15] von Krogh, G., S. Spaeth, and K.R. Lakhani, Community, Joining, and Specialization in Open Source Software Innovation: A Case Study. *Research Policy*, 2003. **32**(7): p. 1217-1241.

[16] Ye, Y., Y. Yamamoto, and K. Nakakoji, A Socio-Technical Framework for Supporting Programmers, in *Proceedings of FSE2007*. 2007. p. 351-360.