# Direct Verbal Communication as a Catalyst of Agile Knowledge Sharing

Grigori Melnik, Frank Maurer
*Department of Computer Science*
*University of Calgary, Calgary (Canada)*
*{melnik, maurer}@cpsc.ucalgary.ca*

## Abstract

*This paper discusses the role of conversation and social interactions as the key element of effective knowledge sharing in an agile process. It also presents the observations made during a repeated experiment on knowledge sharing conducted in various groups of professionals and students. The study suggests that the focus on the pure codified approach is the critical reason of Tayloristic team failure to effectively share knowledge among all stakeholders of a software project. Drawing on the knowledge-as-relationship perspective of knowledge sharing we theorize that verbal face-to-face interaction facilitates achieving higher velocity by software development teams.*

**Keywords:** agile methods, knowledge sharing, verbal communication, conversation, social interaction.

## 1. Introduction

Effective knowledge sharing is key to building a competitive advantage in any organization. Traditionally, two perspectives on knowledge sharing are debated by epistemologists. One, "codification approach", is based on the notion of *knowledge-as-object* (supported by [1], [13], [23], [28]). Such objects can be created, collected, stored, and reused. To be effectively managed and transferred, these objects have to be codified (in the form of documentation, knowledge bases, experience factories, etc.). Another view, "personalization approach", is based on the sociology of knowledge and embraces the notion of *knowledge-as-relationship* ([3], [4], [19], [20]). According to this view, knowledge is uncertain, soft, and embedded in work practices and social relationships.

The view of knowledge-as-objects currently dominates software engineering practices. Documentation[1] is used as the main knowledge transfer[2] medium. Documentation is required by engineering standards at every phase of the software development lifecycle (IEEE, ISO/IEC, PMI, CMM etc.). Some of the persistent problems with such approach are inadequacy, incompleteness, inconsistency, and ambiguity of the written documentation. In a recent investigation of how software engineers use documentation, Lethbridge, Singer, and Forward indicate that almost 70% of subjects (members of Tayloristic[3] teams in 12 corporations and 1 government site) confirmed that documentation is always outdated relative to the current state of a software system [18]. Writing documentation is not easy. Doing so in a clear and unambiguous way is even more difficult. Regardless of the format (prose, diagrams), confusions often arise. In addition, the long knowledge transfer chains with many intermediaries suffer from information distortion and loss and lead to situations when the product delivered is not what the customer really wanted/needed.

Agile methods of software development consider face-to-face interactions (with the customer and among the development team members), "clean code that works", and suites of test drivers as the primary devices for knowledge sharing. The view of knowledge-as-relationship is fundamental. The knowledge is considered to be socially constructed and collectively held. User stories, narratives, and metaphors are viewed as important instruments for knowledge sharing. Yet, documentation is not entirely rejected.

[1] In this paper we refer to process and system documentation, and not to user documentation.

[2] The role of documentation in knowledge preservation of the system is not discussed here.

[3] We prefer to use term "Tayloristic" when discussing traditional, non-agile methodologies. We believe that the latter should not be referred as "plan-driven", because agile methods are also plan-driven. In fact, we would argue that agile methods may involve more planning activities than Tayloristic approaches but of shorter cycles (iterations). The term "task-based" should also be avoided as it points to the side effect of a Tayloristic method, rather than the cause.

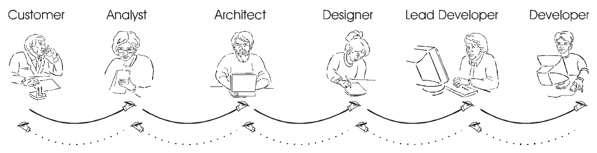
Agilists advocate the use of "lean and mean" documentation when a need is formulated (internally or externally) and documentation satisfies that particular need.

Paraphrasing a Groucho Marx's aphorism – "Outside of a dog, a book is man's best friend; inside of a dog, it's too dark to read" – Ron Jeffries, one of the champions of eXtreme Programming, advises "Outside your extreme programming project, you will probably need documentation: by all means, write it. Inside your project, there is so much verbal communication that you may need very little else." Jeffries concludes "Trust yourselves to know the difference" [15].

## 2. Tayloristic Knowledge Sharing: Knowledge-as-objects

Tayloristic[3] development approaches emphasize strong conformance to a plan through up-front requirements gathering and up-front systems design. In order to control change, knowledge of all possible requirements, design, development, and management issues is externalised to multitudes of documents ("codified") to ensure all issues are first captured and then addressed.

The emphasis on using knowledge-as-objects (documentation in paper or electronic formats) for knowledge sharing is reinforced by practices such as "document what you do" and "do what has been documented" as defined in SEI-CMM [22]. Such practices encourage process groups in an organisation to press development teams to produce multitude of documents throughout a software project.



10% communication error ⇒
$(90\%)^5$ = **59%** info gets to developer

5% communication error ⇒
$(95\%)^5$ = **77%** info gets to developer

**Figure 1. Knowledge Sharing: Tayloristic Way.**

Labour division is often rigorously enforced and specializations dominate. People are deemed to be easily replaceable. Tayloristic methods result in long chains of knowledge transfer (as depicted in Figure 1). Many intermediaries are involved and the original content mutates as it is passed through the conduits of analysts, architects, designers, project leaders etc. The more conduits there are in this chain, the more information is lost or distorted.

For example, let us consider a chain of six players: Customer → Analyst → Architect → Designer → Lead Developer → Developer. Assuming that 5% of communication error occurs at every knowledge transfer, only 77% of the original information gets correctly to the developer. In a similar scenario with 10% of communication error, the portion of the correct information transferred the developer is reduced to 59%. For longer chains, the amount of information is further reduced.

Similar observations are made by Keil and Carmel. In an exploratory study of 31 software development projects, they came to a conclusion that "less successful projects suffered not only from a low number of [customer-developer] links[4] but from a low number of *direct* links". Seventy two (72%) percent of less successful projects involved either zero or just one direct link between customer and developer [16].

Moreover, since people on one end of the chain (information producers) do not know what people on the other end (information consumers) really need, they tend to over-document. This results in even more documentation produced that has practically no value[5]. The more documentation is written in the first place – the more effort will be required to find appropriate information, maintain the documents and keep them up-to-date.

In a Tayloristic process, direct communication between the customer and the developer is discouraged. The communication chain fails to go both ways (depicted as dotted arrows in Figure 1), though selected feedback is sometimes provided.

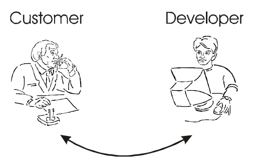
## 3. Agile Knowledge Sharing: Knowledge-as-relationships

In contrast, agile methods are human-centric bodies of software development practices and guidelines that consider individuals and interactions as crucial factors of the project success. Essentially, all

[4] Keil and Carmel define "customer-developer link" as both a channel (i.e. a medium for communication) and a technique (i.e., a method of communication). From a practical standpoint, they chose not to separate these two dimensions because they found that the subjects of their study (managers) themselves do not distinguish between the two dimensions.

[5] Unless each conduit adds specific value that is defined and recognized by the team members and/or the customer.

agile methods encourage continual realignment of development goals with the needs and expectations of the customer. They concentrate on significantly improving communications (among team members and with the customer) and promote continuous feedback. Knowledge formation and sharing is recognized as a highly social (rather than technical/mechanical/formal) process.



10% communication error ⇒
**90%** info gets to developer

5% communication error ⇒
**95%** info gets to developer

**Figure 2. Knowledge Sharing: Agile Way.**

People are no longer treated as "plug-compatible programming units"[6]. Cross-functional teams are used to facilitate better knowledge sharing. The knowledge transfer chains are shortened by direct communication and collaboration (see Figure 2). This ensures what we call "high-velocity knowledge sharing". Drawing on the example from the previous section, a 10% communication loss will result in 90% of information transferred correctly to the developer (compared to 59%) and a 5% communication loss will result in 95% (compared to 77%). Thus, from a communication perspective, direct contact between customer and developer is preferable to indirect because it decreases filtering and distortion that may occur.

It is important for both customer and developer to have a common frame of reference – a common basis of understanding. If this common perspective does not exist, then "shared meaning must be established before mutual understanding can occur." [9]

Unlike Tayloristic managers who trivialize the power of conversation, the agile teams embrace the power of it. Conversation is considered to be the major communication device. An impressive number of theories in software development community ([27], [7], [2]), human-computer interaction (e.g. [21]), management science ([8],[9]) and applied psychology ([10],[29]) speak to the issue of communication media use and, in particular, to the importance of face-to-face conversations. In view of that, practitioners of software development (Weinberg [27]) and agile communities (Cockburn [7], Ambler [2]) independently analyze project experiences, use informal[7] curves and hierarchies of communication media effectiveness/richness (see Figure 3, for example) and unanimously contend that the most effective type is "person-to-person, face-to-face" [7].

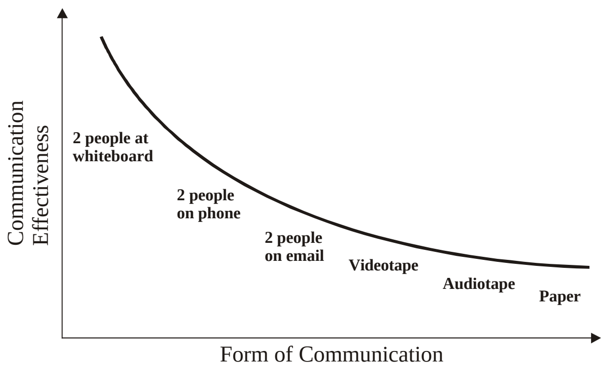


**Figure 3. Alistair Cockburn's Modes of Communication Curve (reproduced from [7])**

In management science, Media Richness Theory ([8], [9]) suggests that direct face-to-face channels offer the prospect of richer communication because of the ability to transmit multiple cues (e.g. voice inflection, body language) and to "facilitate shared meaning" with rapid mutual feedback and personal focus, feelings and emotions infusing the conversation.

Furthermore, the easier it is to communicate, the faster change happens [5]. Via "high-touch" communication (vs. "high-tech" communication in Tayloristic teams) and high-velocity knowledge sharing, agile teams become effective and fast-moving – they manage to deliver the product sooner to the customer. Specifically, in extreme programming, this "high-touch", "warm" interaction takes the form of user stories communicated by conversation, sketches on a whiteboard discussed by the team, code/system knowledge shared during pair programming sessions, and collective code ownership. In Scrum, daily scrums and post-sprint meetings ensure collaborative teamwork and knowledge sharing as the project progresses. Agile Modelling promotes the principle that "Everyone Can Learn from Everyone Else". This principle defines a mindset that enables communication – "someone who believes they can learn something from the person(s) they are communicating with, is

[6] A term coined by Kent Beck and used in [11], for example.

[7] These popular curves are based on project experiences of these practitioners and are not scientifically substantiated.

much more receptive than someone who believes otherwise" [2].

If facts need to be communicated, written documentation, as a medium, can be used (since facts are precise). If concepts, ideas or desires need to communicated, verbal channels are considered to be more effective in facilitating this type of knowledge sharing as they allow rapid mutual feedback.

It is important to recognize that conversation goes beyond simple sharing of information. It stimulates further thinking. As Theodore Zeldin points out, conversation is "a meeting of minds with different memories and habits. When minds meet, they don't just exchange facts: they transform them, reshape them, draw different implications from them, engage in new trains of thought. Conversation doesn't just reshuffle the cards: it creates new cards." (in [29], p.14). Conversation and social interaction help to create common knowledge out of an experience. This common knowledge can be rapidly adjusted, clarified, and reinterpreted.

## 4. Knowledge Sharing Exercise

In this paper we draw upon the experiences of conducting a simple knowledge sharing exercise in the classes we teach and presentations we make. Our observations of the outcomes of these exercises and discussions with participants provided a holistic overview of the knowledge sharing process that we present here.

The participants (97 in total) were asked to form small teams of 6 – 9 people[8]. Further, each team divided itself into three categories of IT workers: *analysts*, *messengers* and *developers*. The simplified job descriptions of each category were explained to the participants and certain time was given for teams to self-organize (at least one person was required for each subteam; however, teams were allowed to freely decide how many analysts, messengers and developers they would need). *Analyst's* job involved describing requirements in prose on paper. *Messengers* carried notes from analysts to developers and back. They were not allowed to communicate in any way with other groups. *Developers* were responsible for implementing the specification. *Analysts* and *Developers* were separated in various locations (rooms, hallways etc.) so that they could not overhear conversations of the other subteams. Subjects were not allowed to swap jobs during the exercise.

The task was simple enough for the participants to perform – specify a sample drawing so that the developers could reproduce it. Examples of such drawings are shown in Figures 4 and 5. Messages could only contain prose in English, in usual left-to-right multi-line format. Teams were given 20 minutes to accomplish the task. In all experiments flip chart paper (27x34") was used, except for two teams of graduate students who used colored Letter format paper. Assigned drawings were created by the authors of the paper and contained two or three abstract figures (see Figures 4 and 5, section on the left-hand side[9]). In one experiment the sample drawings (Figure 4d) were more structured and symmetrical than in others.

Clearly, in the academic environment the assumption of subjects having a similar background is met. This was not guaranteed with the teams of practitioners due to the randomness of team distribution.

One critical point of view on the exercise might be that drawing abstract figures does not adequately reflect the activities undertaken during software development. Although our exercise is intentionally simplified, we argue that it does resemble enough of such activities to be examined. Both software development and the act of drawing pictures are mainly about recognizing patterns and creating their representations – coded or visual. "A program is constructed from some basic patterns, and the construction rules can in turn be expressed as other types of patterns"[10] [6] In both activities (drawing and developing software) patterns need to be recognized, externalized for sharing and finally applied. Therefore, the authors believe the described exercise is appropriate.

## 5. Study Subjects

Twenty eight computer professionals employed in Calgary, Alberta area were subjects of this study. In particular, 50% identified themselves as software developers, 25% as project managers, and remaining 25% as system administrators. These practitioners attended a presentation on agile methods given by the first author at a regular meeting of the Calgary Unix Users Group (CUUG). Four teams were formed, one of

[8] In all cases, except for the University graduate class, subjects were allowed to choose their own teammates. Graduate students were assigned randomly, based on their seating arrangement in the auditorium.

[9] Figures reproduced in this paper were captured with a digital camera and then traced into black-and-white line-art for publication purposes.

[10] Thus, a special subject called Design Patterns.

which consisted entirely of the employees from the same company who knew each other well.

In addition, students of three different levels – (1) College-level Post-Diploma Applied Bachelor's[11], (2) University B.Sc., and (3) University M.Sc. – of computer science programs from the University of Calgary and the Southern Alberta Institute of Technology (SAIT) were the subjects of this exercise. All individuals were knowledgeable about requirements engineering, systems analysis and programming (no junior students were involved). Two thirds of the graduate students had prior industrial experience. SAIT students had 4 teams formed; University undergraduates – 4 teams, and University graduates – 2 teams. The total number of subjects (practitioners and students) was 97 with 14 teams formed.

## 6. Common Observations and Discussion

Overall, our observations were consistent across both industry teams and student teams. Therefore, the following discussion refers to all types of teams. Whenever significant differences occur, we emphasize those individually.

- *Typical role distribution*

Teams opted to the following role distribution: 1 messenger, 2–4 analysts, 3–5 developers. In several cases, a team would have a lead "developer" in charge of drawing. We observed only two development teams (one from industry and one from the undergraduate class) subdividing the task and having subteams of developers work on figures simultaneously. The rest performed "gang development" working on one figure at a time.

- *Inability to complete the task on time*

Nine teams did not manage to complete the task on time. In fact, two teams ended up with lots of written specification but no drawing produced at all. During the post-exercise discussion, the teams recognized the fact that analysts spent most of the time analyzing the drawing and creating the first set of specifications leaving no time for developers to implement it. This invites an analogy with the situation in software industry – where in 2002 roughly 66% of the projects failed or were challenged (according to the CHAOS Report [25]). Often no working code is produced even though lots of comprehensive specification is written causing a situation known as "analysis-paralysis".

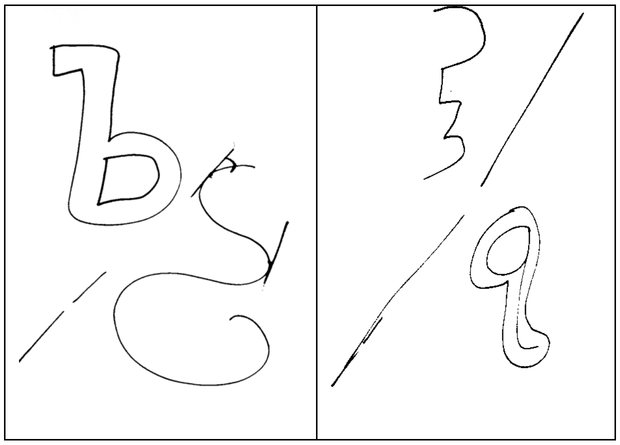
[11] Similar to a B.Tech degree offered by some U.S. polytechnic institutions and colleges.

- *Quality of produced result depends on the complexity of the original*

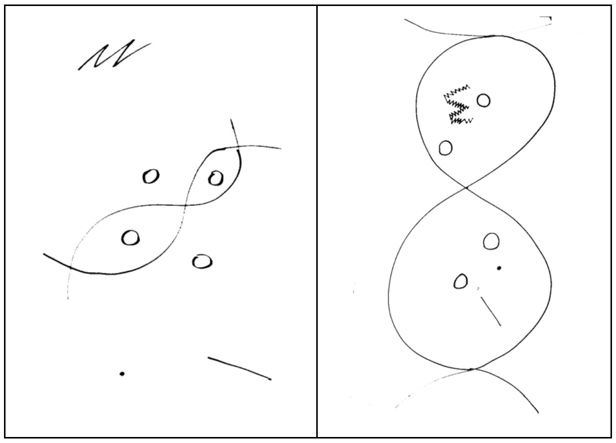
The level of complexity and abstractionism was a significant factor in the exercise. Most of the complex abstract curves (see bottom curves of Figure 4a and 4c, top design in Figure 5a and in the middle of Figure 5c) were not attempted. Only two teams tried to break the complicated curves into simpler pieces and transformations. One team attempted to follow the abstract curve as a path with the instructions to turn left, right, do a half-loop, turn 90 degrees and so on. However, when a more structured design was offered, the teams managed to produce a drawing very close to the one required (see Figure 4d, for example). This observation is supported by the Media Richness Theory postulate that certain media are better to transmit information depending upon whether the information is used in situations of uncertainty and equivocality ([8],[9]). Media are characterized as high or low in "richness" based on their capacity to facilitate shared meaning. A rich medium facilitates insight and rapid understanding. In case of abstract drawings, the degrees of uncertainty and equivocality were quite high (based on the comments of the experiment subjects). As a result, a medium providing higher "richness" was needed. Media Richness Theory ranks face-to-face communication as the "richest" medium. Research in the organization communication suggests that people express themselves more naturally and less formally when speaking (relative to writing). This was also confirmed during the post-experiment informal group discussions: one participant exclaimed, "We would have done a better job if we were allowed to converse".

- *Time estimates were not negotiated by the teams*

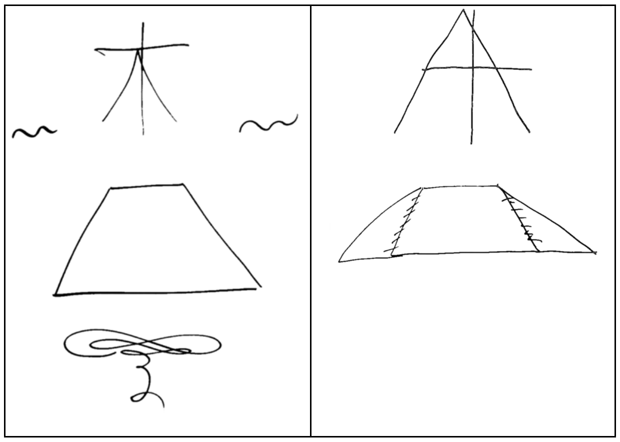
None of the teams attempted to negotiate the time allotted or the scope of the task. Certainly, in an academic setting the students are not accustomed to negotiating time given for the assignment. However, similar observations are made with regard to industrial teams (and not only during our experiment). As Watts Humphrey points out, "Most engineers are so focused on the job that they don't think about what the manager is saying. When managers say the delivery date is nine months, they are making a bid. And you [developers] buy it without a counter offer. You'd never buy a house or a car or a boat this way. You'd debate the number" [14]. Our study shows that practitioners did not even attempt to "debate the number". They took on the task without proper evaluation of its complexity. Several participants mentioned at the end of the exercise that they knew it would be impossible to complete the task.


SAIT Team 1: Original SAIT Team 1: Product

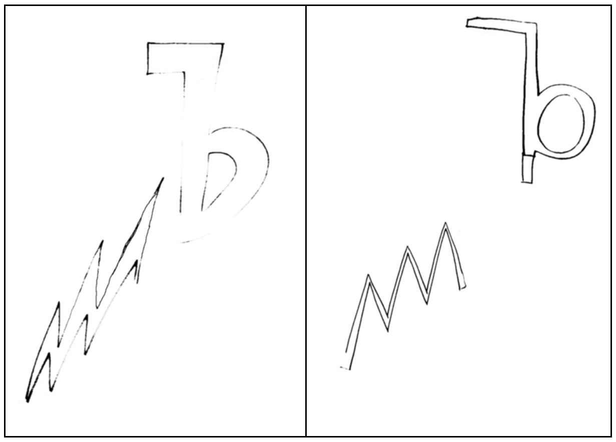
**(a)**


SAIT Team 2: Original SAIT Team 2: Product

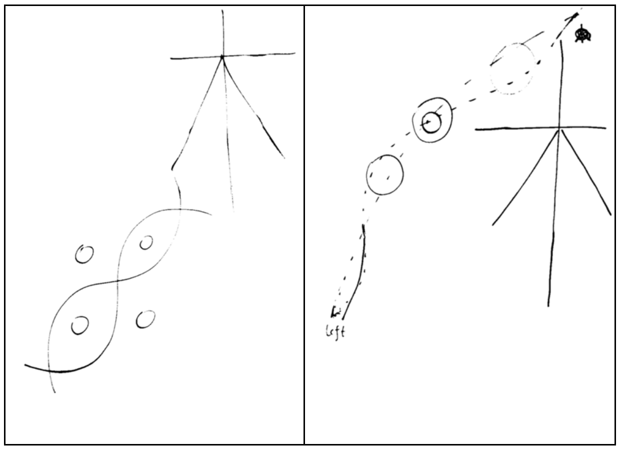
**(b)**


SAIT Team 3: Original SAIT Team 3: Product

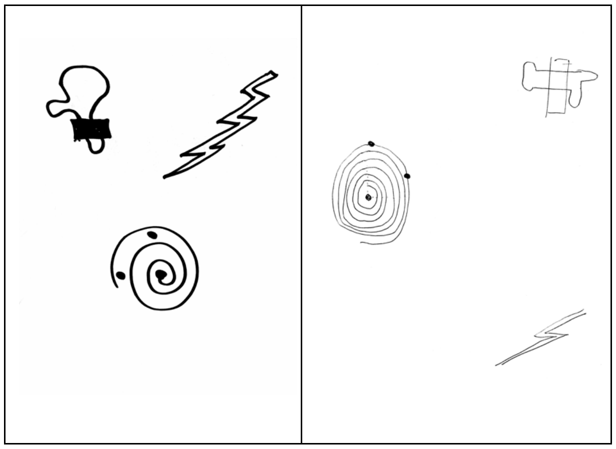
**(c)**


UofC Team 1: Original UofC Team 1: Product

**(d)**

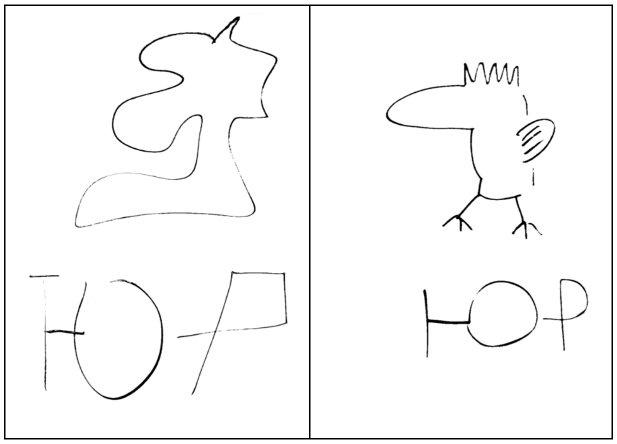

UofC Team 2: Original UofC Team 2: Product

**(e)**

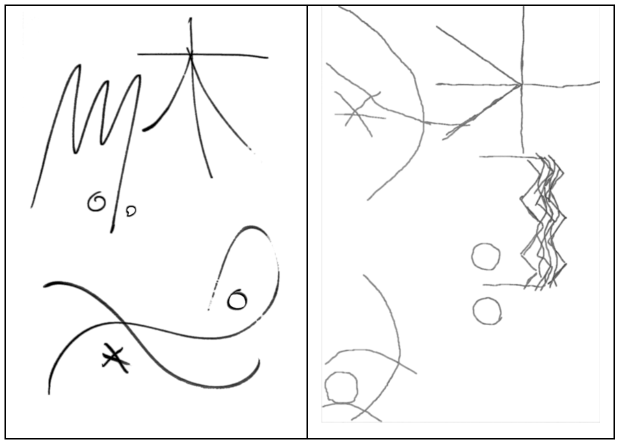

UofC Grad Team 3: Original UofC Grad Team 3: Product

**(f)**

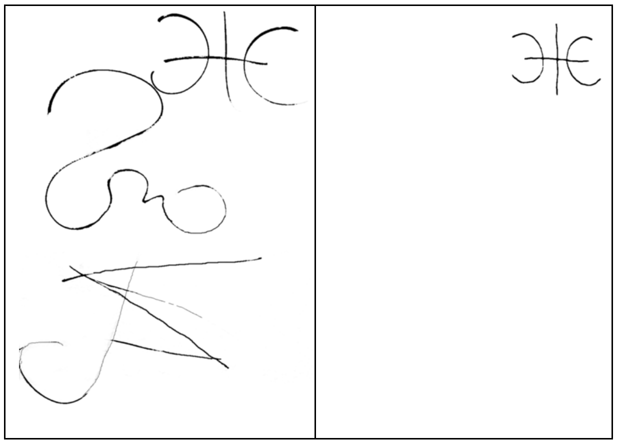
**Figure 4. Selected Original Drawings and Results Produced by Student Teams.**


Industry Team 1: Original Industry Team 1: Product

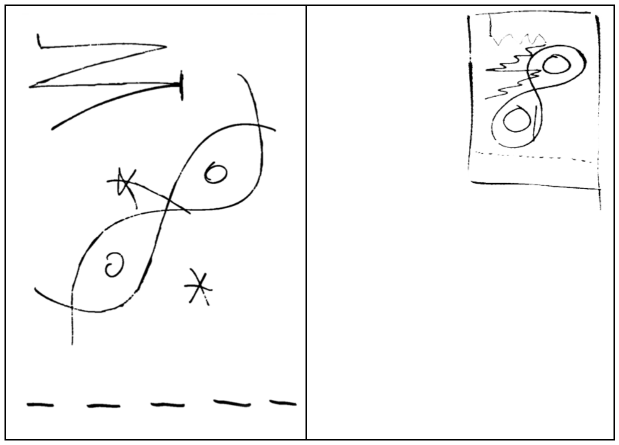
**(a)**


Industry Team 2: Original Industry Team 2: Product

**(b)**


Industry Team 3: Original Industry Team 3: Product

**(c)**


Industry Team 4: Original Industry Team 4: Product

**(d)**

**Figure 5. Original Drawings and Results Produced by Industry Teams**

▣ *Written documentation is not easy to produce*
Many participants struggled with specifying requirements using natural language. Their descriptions suffered from noise (information irrelevant to the problem), silence (omission of important aspects of the problem), and equivocality (several possible interpretations of the same phrase). Consider a sample communication log depicted in Figure 6. The original drawing and the product are shown in Figure 5c. The first batch of specifications written by analysts suffers from all three "sins": noise ("Think Abstract"-"About What?"-"Never Mind"), silence (no indication of where the digit 8 starts; numerous clarifications are needed), and ambiguity ("ski-jump"; "East"). The written specification simply does not convey enough information for developers to perform their task. Another example of equivocality and misinterpretation can be seen in the specification of team 5d (Figures 7 and 5d). Their indication of the "portrait orientation, approx 8½ x 11 ratio" was interpreted by the developers as explicit indication of the dimensions of the drawing. This resulted in designating a small area of the given flipchart for drawing. As evident from additional three logs (Figures 7–9) included in the paper, requirements specifications written by other teams suffer from similar flaws[12]. Analyzing Figure 8, the phrase "looks like a tree" is clearly ambiguous. "What kind of tree?" developers ask. "Is it leafy or just branches?" There are numerous shapes of trees and the above specification only confuses them. Analysts struggle with finding the right words. They discard the

[12] Reader is invited to attempt drawing the figures based on the specifications provided. Compare your results with the original drawings.

tree idea and try another one, which, apparently, is no better – "squiggly head with a horn". Developers are puzzled. They request clarification. However, instead, they receive the third description – "Woodstock [character] of Peanuts [cartoon]". Unsurprisingly, the result is not what was desired (see the top half of the Product, Figure 5a).

▣ *Written documentation has low communication bandwidth*

The process of sharing knowledge in written language is time and effort consuming. Something that could have been easily discussed in a conversation takes longer to transmit via written documentation. As cognitive science indicates, the translation of thought to speech is much faster then the transition of thoughts to writing [12]. Moreover, this translation is less cognitively demanding [17]. Our evidence also suggests that knowledge sharing is more intensive in the direction from analysts to developers. Generally, developers are discouraged from communicating to the customer. There were even three development teams in our study that have never communicated back to the analysts (see Figure 9, for example). When asked why, they believe, the teams failed to produce satisfactory results, practitioners and students recognized that written communication was hard and did not allow timely adjustments of the messages communicated.

▣ *Important details are overlooked*

Admittedly, analysts tend to overlook the details that are important to the customer. For example, in our

| **Analysts Specify:** | **Developers Clarify:** | **Analysts Respond:** |
|---|---|---|
| Start with making a large digit 8… | Start where? at top? go clockwise on upper circle? | Yes |
| ...stop at 40%... | of length? | Yes |
| Without lifting your pen start a smaller 8, stop at 20%... | start where? at top? | Yes |
|  | go clockwise on upper circle | Yes |
| without lifting your pen go 1.5 in East | East? Do you mean right? | Yes |
| make a full teardrop (counter clockwise) (@170°) | start where? oriented how? |  |
| ski-jump (18" inc's) | start where, at top? |  |
| with a almost enclosed loop on bottom | OK. |  |
|  | Where do we put this? |  |
| Think abstract | About what? | Never mind |

**Figure 6. Sample Communication Log[13] (Team 5c).**

| **Analysts Specify:** | **Developers Clarify:** | **Analysts Respond:** |
|---|---|---|
| (portrait orientation, approx 8 1/2 x 11 ratio) |  |  |
| The figure consists of 3 major components: |  |  |
| at bottom, five dashes spanning width of image (approx 1/10 of the way up from bottom) | One component has been described. What are the other two (2) components? |  |
| top left | What the heck is this? |  |
| zigzag: in approx 10% from the top left draw a line to center of page (same height), continue back to left, but down approx 10% of page height |  |  |
| continue: down right, to center (same height) |  |  |
| continue left and down (as above). |  |  |
| add to zigzag: at top left corner, short vertical line extending upward (approx same length as dashes) | ? |  |
| add to zigzag: right Vertex: short vertical line centered on vertex |  |  |
| middle 2/3 of page: |  |  |
| large figure eight rotated 45° clockwise, drawn as like intersecting sine waver 90° of phase, so a bit of each wave extends part the intersecting figure eight. |  |  |
| with each loop of the eight, a circle approx half radius of loop |  |  |

**Figure 7. Sample Communication Log[13] (Team 5d).**

[13] Verbatim, without correction of spelling or grammatical errors.

| Analysts Specify: | Developers Clarify: | Analysts Respond: |
|---|---|---|
| Top Picture: | | Top Picture: |
| - looks like tree (basic shape) | What kind of tree? | - looks like 'woodstock' (Peanuts) in profile |
| - continuous line around outside | fur, leafy (apple tree), or just branches? | - toe to left of picture |
| - 'Nose' to left | | - never mind tree |
| - Foot shape at bottom | | - describes whole |
| - Squiggly head with horn at top | - sqigly head on top of rest of picture or does it describe whole picture? | |
| - Looks like profile | | |
| Bottom Picture | RAN OUT OF TIME… | |
| - right side of 'H' is open at 11:00 in counter clockwize circle | | |
| - 'P' is offset with bottom line extended through stem like a kite in appearance | | |

**Figure 8. Sample Communication Log[13] (Team 5a).**

| Analysts Specify: | Developers Clarify: | Analysts Respond: |
|---|---|---|
| In Top Right Corner of sheet | | |
| - Chinese character for wood | | |
| - Height is half the vertical legth of sheet | | |
| - width is half the horizontal length of sheet | | |
| - vertical line in character extended | | |
| In the bottom left section. Two diagonal intersecting curve from buttom left to top right. These two intersecting line intersect 3 times. In the middle intersection is surround by two individual circle | | Additional Info: In the 2 area enclosed by these 2 intersecting lines, there are 1 circle per each area. |

**Figure 9. Sample Communication Log[13] (Team 4e).**

study none of the teams (with the exception of one) produced the drawing using the correct color. It was never explicitly stated whether specific colors needed to be used. Teams were allowed to pick markers of any color. However, without any clarification or consultation with the client, teams simply assumed the color did not matter. Another example when details were overlooked is in the selection of paper orientation. Two teams produced the results upside down (e.g. Figure 4a) and one other team produced the drawing rotated at 90 degrees clockwise (Figure 5b). These situations are notoriously common in industry when analysts or surrogate customers (possibly supervisors, internal or external consultants etc.) are used for specifying requirements – they often perceive the needs of the real customer differently and, as a result, specifications produced are not aligned with true customer needs.

▣ *Periods of inactivity cost money*

While the analysts were busy describing the figures in natural language, the developers were idle. This, in fact, was observed not only during the initial stage of specifying the first set of requirements, but during the whole exercise. These "intermissions" are usually not long enough for the developers to engage in other development activities.

▣ *Knowledge of subject domain helps*

Without doubt, knowledge of the subject domain helps in both requirements specification and their implementation. The teams that could deduce in the figures specific meanings that were known to both analysts and developers did very well with implementing those figures. For example, a Japanese hieroglyph for a tree was used in one of the drawings (Figures 4c, 4e, 5b), and a Cyrillic character "?  " was utilized in Figure 4a. Of course, this requires common knowledge to be present. If the analyst described a figure as a Japanese hieroglyph for a tree, then the developers need to know what it looks like, otherwise that information will have no value to them. For that reason, emergent common knowledge as well as insight in the background knowledge of the receiver of the shared information plays an important role in effective communication.

▣ *The customer is never consulted*

Evidently, the customer was never asked questions about the original drawing or the task that the teams were assigned to perform. Although the customer was present and available at all times, teams proceeded to act based on initial instructions provided. There seemed to be no need to interact with the customer. This is very typical in Tayloristic processes.

## 7. Conclusions

Since knowledge is the most strategic resource in today's world, effective knowledge sharing is imperative for a software engineering team to succeed. To achieve that, effective communication, which involves both content and relationship dimensions, is required. The pure codified view of knowledge-as-object cultivates the use of externalized knowledge in the form of written documentation (predominantly in Tayloristic teams). Our exercises collectively demonstrate ineffectiveness of such knowledge sharing when complex cognitive artifacts are used. We believe the higher is the level of abstractionism (complexity), the more is the need for interactive knowledge sharing via direct verbal communication. These findings are drawn upon our experiment, the arguments of the Media Richness Theory and research in Cognitive Psychology. Face-to-face channels offer the prospect of richer communication because of the ability to transmit multiple cues (e.g. physical presence, voice inflection, and body language). Such direct links are particularly important when there are high levels of equivocality (ambiguity) and uncertainty – situations especially likely to occur in communication of requirements. It is important to recognize that any communication, formal or informal, requires common knowledge in order to adequately interpret messages communicated. In addition, it is also important to be aware of the background knowledge of the information recipient to streamline what needs to be shared and what can be omitted in the communication.

Finally, one additional argument is a speculative one, based on the study of customer-developer links in software development by Keil and Carmel [16]. We believe the shortened knowledge transfer chain in an agile process results in high-velocity accomplishments and the success of the software development project. In contrast, long chains of intermediaries result in ineffective communication between customer and developer, because intermediaries filter and distort messages (mostly unintentionally) and they may not have a complete understanding of customer needs.

## 8. Future Work

There is a great deal of future research that needs to be conducted on communications and knowledge sharing in software teams. Although our current exercise looked in detail at knowledge sharing in Tayloristic teams, it does not empirically address the issue of the effectiveness of communication among agile team members. Additional experiments with various subjects are planned. We intend to look at how effectiveness of communication changes in industry work groups, if analysts were allowed to preview the work of developers (as was suggested by one of the participating teams). Also, we will look deeper in the psychology and team dynamics – a similar exercise can be performed with established teams (for people who worked together for some time). Furthermore, we plan to perform experiments involving knowledge transfer chains with various numbers of intermediaries.

We hope that our observations will provoke discussion and future studies on a wider selection of subjects and would like to invite any interested parties to take part in the future experiments.

## Acknowledgements

The authors would like to acknowledge the great insights and help of Michael Richter, Marquis Bureau, Vladimir Melnik, and Kristopher Read. In addition, we thank all students and practitioners, who took part in the exercise and provided us with their valuable feedback. Current research was partially supported by the National Sciences and Engineering Research Council of Canada (NSERC) and the Alberta Informatics Circle of Research Excellence (iCore).

## References

[1] Alavi, M., Leidner, D. Knowledge Management Systems: Issues, Challenges, and Benefits, *Communications of the AIS*, 1(7): 2–36, 1999.

[2] Ambler, S. Communication. The Essay. Online: http://www.agilemodeling.com/essays/communication.htm Last accessed Feb 14, 2004.

[3] Boland, R., Tenkasi, R. Perspective Making and Perspective Taking in Communities of Knowing, *Organization Science*, 6(4): 350–372, 1995.

[4] Brown, J., Duguid, P. *The Social Life of Information*. Harvard Business School Press, Boston, MA, 2000.

[5] Burke, J. *Connections*. Little Brown & Co, New York, NY, 1995.

[6] Chang, S. *Handbook of Software Engineering and Knowledge Engineering*, Vol. 1. World Scientific, River

Edge, NJ: vi, 2001.

[7] Cockburn, A. Characterizing People as Non-Linear, First-Order Components in Software Development. *Proc. 4th International Multi-Conference on Systems, Cybernetics, and Informatics, Orlando, FL*, Vol.1. International Institute of Informatics and Systemics, Skokie, IL, 2000.

[8] Daft, R., Lengel, R. Organizational Information Requirements, Media Richness and Structural Design. *Management Science*, 32(5):554–571, 1986.

[9] Daft, R., Lengel, R., Trevino, L. Message Equivocality, Media Selection, and Manager Performance: Implications for Information Systems. *MIS Quarterly*, 11(3): 355-366, 1987.

[10] Doherty-Sneddon, G., Anderson, A., O'Malley, C., Langton, S., Garrod, S., Bruce, V. Face-to-face and Video Mediated Communication: a Comparison of Dialogue Structure and Task Performance. *Journal of Experimental Psychology (Applied)*, 3(2):105-125, 1997.

[11] Fowler, M. The New Methodology. ThoughtWorks, Inc., Chicago, IL, 2000. Online: http://www.thoughtworks.com/library/newMethodology.pdf. Last accessed Feb 1, 2004.

[12] Gould, J. An Experimental Study of Writing, Dictating, and Speaking. In Requin, J. (ed.), *Attention and Performance VII*. Lawrence Earlbaum, Hillsdale, NJ, 1978.

[13] Hansen, M., Haas, M. Competing for Attention in Knowledge Markets: Electronic Document Dissemination in a Management Consulting Company, *Administrative Science Quarterly*, 46(1): 1–28, 2001.

[14] Humphrey, W. *The Watts New? Collection*. Carnegie Mellon Software Engineering Institute, Pittsburgh, PA, 1998, p.13.

[15] Jeffries, R. Essential XP: Documentation. Online: http://www.xprogramming.com/xpmag/expDocumentationInXP.htm . Last accessed January 28, 2004.

[16] Keil, M., Carmel, E. Customer-Developer Links in Software Development. *Communications of the ACM*, 38 (5): 33–44, 1995.

[17] Kroll, B. Cognitive Egocentrism and the Problem of Audience Awareness in Written Discourse. *Research in the Teaching of English*, 12: 269–281, 1978.

[18] Lethbridge, T., Singer, J., Forward, A. How Software Engineers Use Documentation: The State of the Practice, *IEEE Software*, 20(6): 35–39, 2003.

[19] Nidumolu, S., Subramani, M., Aldrich, A. Situated Learning and the Situated Knowledge Web, *Journal of Management Information Systems*, 18(1): 115–150, 2001.

[20] Nonaka, I., Konno, N. The Concept of "Ba": Building a foundation for Knowledge Creation, *California Management Review*, 40(3): 40–55, 1998.

[21] O'Conaill, B., Whittaker, S., Wilbur, S. Conversations Over Videoconferences: an Evaluation of the Spoken Aspects of Video Mediated Interaction. *Human Computer Interaction*, 8(4): 389-428, 1993.

[22] Paulk, M., Curtis, B., Chrissis, M., Weber, C. Capability Maturity Model, Version 1.1. *IEEE Software*, 10(4): 18–27, 1993.

[23] Szulanski, G. The Process of Knowledge Transfer: A Diachronic Analysis of Stickiness, *Organizational Behavior and Human Decision Processes*, 82(1): 9–27, 2000.

[24] Taylor, F. *Principles of Scientific Management*. Norton, New York, NY, 1967.

[25] *The CHAOS Chronicles*. The Standish Group International, West Yarmouth, MA. Online http://www1.standishgroup.com//chaos/intro2.php. Last accessed January 20, 2004.

[26] Watzlawick, P., Beavin, J., Jackson, D. *Pragmatics of Group Communication: A Study of Interactional Patterns, Pathologies, and Paradoxes*. Norton, New York, NY, 1967.

[27] Weinberg, J. *The Psychology of Computer Programming*, Dorset House, New York, NY, 1998.

[28] Zack, M. Managing Codified knowledge, *Sloan Management Review*, 40(4): 45–58, 1999.

[29] Zeldin, T. *Conversation*. Harvill Press, London, UK, 1998.