

A Proposal of TIE Model for Communication in Software Development process

Masakazu Kanbe^{1,2}, Shuichiro Yamamoto¹ and Toshizumi Ohta²

¹ Research Institute for System Science, NTT DATA CORPORATION,
3-3-9 Toyosu Koutoku Tokyo, Japan

² Graduate Department of Social Intelligence and Informatics,
Graduate School of Information Systems, The University of Electro-Communications,
1-5-1 Choufugaoka, Choufushi, Tokyo Japan

kanbems@nttdata.co.jp, yamamotosui@nttdata.co.jp, ohta@is.uec.ac.jp

Abstract. Communication is more important in software development fields. We proposed the intermediary knowledge model to analyze the enterprise communication by extending traditional knowledge creation model. In this article, we propose TIE models based on intermediary knowledge. TIE model is the knowledge network model to explain the just in time documentation in the CMC tools like wiki. We analyzed the case of wiki based software development and showed the effectiveness and efficiency of the CMC tools in software development in certain conditions.

Keywords: Knowledge communication, Software development, Knowledge network

1 Introduction

Software developments become more complex and lots of people, which has various backgrounds participant in its processes. Various communications occurred in the field of the software development. The communication style of software developments contains regular meetings, ad hoc conversations in the local office, and acceptances of document by e-mail. Furthermore, CMC (Computer Mediated Communication) tools such as wiki, SNS, blogs and communication plug-ins of Integrated Development Environment support the developers' communication. In this article, we propose TIE model as knowledge network model for software development communication. TIE model is three layered network model. The three layers are tacit knowledge network, intermediary knowledge network and explicit knowledge network. We analyzed the case of wiki used software development process by TIE model. We also investigate the effectiveness and efficiency of wiki used software development process.

2 Related Works

In this chapter, we introduce the previous related works to explain our model.

2.1 Intermediary knowledge model

We proposed the intermediary knowledge model as knowledge sharing model in enterprises [1] [2]. Intermediary knowledge is the knowledge statement in which employees share the knowledge by the CMC tools. Intermediary knowledge model is one of the extended models of tacit and explicit knowledge concept [3].

The traditional knowledge creation model has tacit and explicit knowledge and four knowledge transformation modes; socialization, externalization, combination and internalization [4]. Intermediary knowledge model is proposed to solve problems and perform business tasks without the knowledge spirals of the organizational process. Employees can share the knowledge in intermediary knowledge statement by using CMC tools. In intermediary knowledge model, employees can exchange the knowledge that cannot be shared in tacit knowledge with less cost compared to explicit knowledge.

Fig. 1 shows the intermediary knowledge model. The dashed lined square in Fig. 1 indicates the traditional knowledge creation model. We add the intermediary knowledge to the traditional model. This model indicates that the employees can rapidly develop the knowledge in the CMC tools by using the intermediary knowledge transformation modes. The modes consist of publication, fragmentation, collaboration, resonant formation, and sophistication. Publication means to publish individual experience or ideas. Fragmentation means to import the parts of explicit knowledge. Collaboration means to react with employees' problems or opinions. Resonant formation means to accept and understand the others' opinions. Sophistication means to develop explicit knowledge from intermediary knowledge.

According to the traditional knowledge spiral condition, if employees intend to use the knowledge formally and inter-organizationally, each organization have to generate the explicit knowledge through the inner organizational knowledge spirals. Formally making explicit knowledge needs high cost and much labor through the inner organizational knowledge spirals. Intermediary knowledge transformation modes explain lower cost and labor in the knowledge exchange than the traditional knowledge transformation modes.

Also intermediary knowledge model explains the effectiveness of communication records. CMC tools create the more interaction points for employees than they do not use CMC tools. The employees have new communication in CMC tools. The communication in CMC tools are recorded as intermediary knowledge and employees reused the knowledge efficiently.

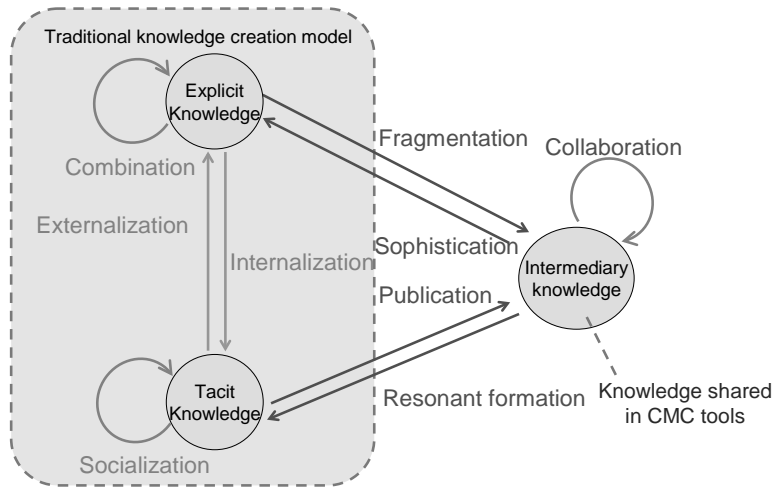


Fig. 1 Intermediary knowledge model

2.2 IBIS model

IBIS [5] and gIBIS [6] [7] are the traditional software engineering method. One of the purposes of these IBIS methods is to fully record and structuralize the discussion processes and progresses in software developments. Recording and structuring all the discussion processes and progresses, software developers could find the important information in developments. Although the records and structures of IBIS or gIBIS may be useful, the costs or labors are too large to make and reuse the full documented records.

2.3 Recent software engineering researches

Software engineering researches supports the software developers' communication. Software developments need the knowledge communication among software developers. Ko et al. [8] analyzed the software developers' activities and found that the developers used coworker as information source. This research indicates that the communication among the developers is very important in recent software developments.

Ye et al. [9] [10] helped the software developers to search the knowledge from the software libraries and members of software development team. They proposed the personalized search engine for API documents and communication channels for experts in software development team. Their researches implicate the way to make developers communicate each other for efficient software development.

Marczak et al. [11] indicated the importance of information brokers in requirement change management. As their research, the information brokers have important roles in the social network of software development team. The information brokers

facilitate information flow to avoid misinterpretations of requirements. These researches indicate the importance of the developers' communication in software development.

3 TIE model

3.1 Overview of TIE model

We propose TIE model as CMC model for dynamic communication in software development process. TIE model has three layers consisted of Tacit Knowledge Network (TKN), Intermediary Knowledge Network (IKN) and Explicit Knowledge Network (EKN). Table 1 shows features of these three layers. Table 1 indicates the definitions of TIE model.

Table 1. Features of layers of TIE model

Knowledge Network	Network node	Media	Documentation	Examples of products
Tacit Knowledge Network	Human	Face to Face, Telephone, Video conference	No documentation	Discussions, Meetings
Intermediary Knowledge Network	CMC content	CMC tools	Just in time documentation	CMC logs
Explicit Knowledge Network	Document	Document management services	Full documentation	Requirements, Specifications, Source codes, Manuals

TKN has roles to exchange the tacit knowledge. The network node of TKN is human. TKN is related to organization structures, roles of members, processes of decision making and so on. TKN is occurred in face to face meeting, telephone or video conference communication. It seems that TKN brings down no documentation for the software development. We assume TKN does not create any formal documents. The products of TKN are discussions and meetings. TKN does not always create the tangible products to be observed.

IKN has roles to exchange the intermediary knowledge. The network node of IKN is CMC content. IKN is related to CMC network in the software development team. These CMC contents grow up in CMC tools such as Wiki, Blog, and SNS. IKN provides just in time documentation with the developers. If one needs to coordinate with others, one can use CMC tools to coordinate with others. And the coordination

records are published for all the members of software developmental teams. These published coordination records are useful documents for software development. We call this process “Just in time documentation.” Just in time documentation means that the necessary knowledge becomes documents when the developers communicate each other in CMC tools. The products of IKN are CMC logs.

EKN has roles to exchange the explicit knowledge. EKN is related to document network in the software development process. The network node of EKN is document. This document network grows up in document management services, which of functions are the document traceability, the historical management, the full text search and the file sharing. EKN provides full documentation with developers. The products of EKN are documents, such as requirements, specifications, source codes, manuals and guidelines.

The network edges of TKN mean human oral communication. The edges of IKN mean the concatenations of CMC contexts. The edges of EKN mean the interrelationship among documents. The edges between TKN and IKN mean the processes of intermediary knowledge provisions and acquisitions via CMC tools. The edges between IKN and EKN mean the processes of quotations and documentations of explicit knowledge via CMC tools.

3.2 TIE model for software development communication

TKN do not create any formal document. We call this TKN statement “no documentation.” EKN aims to create the documents elaborately. We call this EKN statement “full documentation.” Traditional software developments use the TKN and EKN as the knowledge process.

However, the knowledge processes in the traditional software developments has two problems. First problem is the loss of the important information of software developments. The knowledge processes in TKN are communication in discussions and meetings. Communication records of TKN are almost disappeared when the meetings or discussions ended. Communication contents of TKN are not always described in documents and merely shared with all the members.

Second problem is the difficulty to record the all the important information of software developments. If all the events in software developments were documented fully at right time, each member could understand requirements, specification, and source codes perfectly. However, it is difficult to achieve full documentation because its cost is very high and its range is very ambiguous.

Fig.2 shows TIE model we proposed. TIE model adds IKN to the traditional knowledge process in software developments. CMC tools support IKN. Balloons express the representative knowledge process of TIE model. The square balloon means the knowledge processes of traditional software development style. Round balloon means the knowledge processes of particular TIE model.

The knowledge processes in IKN are open and agile communication on CMC tools. The open CMC tools facilitate the communication of software development teams. IKN records the CMC logs. These CMC logs are not formal document, but very useful knowledge for software development. The development team members can read the knowledge processes each other in CMC tools. The knowledge processes in

TIE model have correspondence relation with the knowledge transfer modes in the intermediary knowledge model in Fig. 1. Tacit knowledge, intermediary knowledge and explicit knowledge are corresponded with TKN, IKN and EKN respectively.

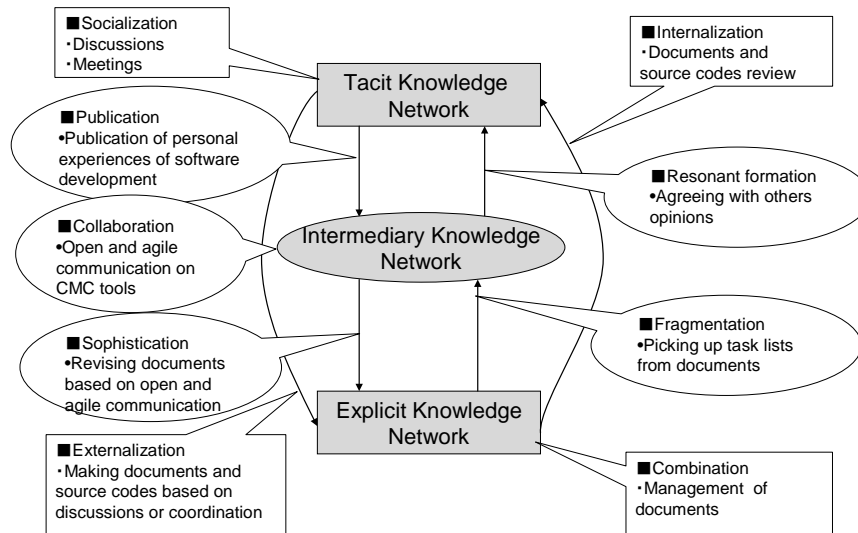


Fig.2 TIE model and knowledge processes

4 Case Study of TIE Model for Software Development

To confirm the effectiveness and efficiency of TIE model, we analyzed the case of the wiki based software development. Wiki was used to facilitate the communication in software development case.

4.1 Aim of the case study

The case study aims to verify assumptions below;

- A1: CMC tools can record the important knowledge for the software development.
- A2: CMC tools can facilitate to share knowledge which did not efficiently share in traditional software development style.
- A3: CMC tool can provide the appropriate communication occasions.

These assumptions are set to confirm the effectiveness and efficiency of CMC tools in software development. We define that to verify these assumptions are to verify the effectiveness and efficiency of TIE model in software development communication because CMC tools support IKN.

4.2 Overview of case

We selected the wiki based software development case. The number of software developers in this case was nine. Nine members belonged to two other companies. They cooperated to develop the software system development with unfamiliar devices. This software development had the processes; document production, program coding, and program test. These two companies office were in different location with no time zone difference. Although the members had the face to face meetings at once a week regularly, some communication mistakes occurred and caused negative effects for the software development. To deal with the communication mistakes, this software development team determined to use wiki to compliment with team communication. Appendix shows the outlines of the wiki communication. This wiki had 13 pages. There were 21 items in 13 wiki pages.

We observed the knowledge processes of TIE model, 18 publications, 5 fragmentations and 2 collaborations in wiki. An example of the publication was W3 in Appendix. The case of W3 was that a member imagined the tasks to be done for this process and published them. An example of fragmentation was W12 in Appendix. The case of W12 was that a member extracted documents lists from the development conference regulations in this organization. An example of collaboration was W8 in Appendix. The case of W8 was that members communicated about the test policy in wiki. We did not observe the knowledge process of resonant formation and sophistication. These two knowledge processes might occur outside wiki in this case.

4.3 Verifying the assumptions

To verify the assumptions, we picked up the evidences from the CMC in the wiki.

- Verifying A1: CMC tools can record the important information for the software development.

This assumption is related to “recording.” The wiki recorded the important information for the software development. Nine members used the wiki and recorded the 21 important knowledge items for the software development. For example, there were design documents lists in W1 and W2 of Appendix. These lists were fragmented intermediary knowledge from the explicit knowledge such as document inventories. A member considered that it is necessary for the software development team to share the documents lists. This member extracted the document names from the document inventories.

This member did not use e-mail to share the document lists, but used wiki. Other members added the progress information with the document items in wiki. To add the progress information is publication of intermediary knowledge. All the member shared the dairy progress information of each document by wiki. The knowledge in the wiki is open to all the members and shared one target. If they shared this progress information by e-mail, they would not continue to refine the progress information because e-mail has the feature of cross in the post.

- Verifying A2: CMC tools can facilitate to share information which did not efficiently share in traditional software development style.

This assumption is related to “effective knowledge sharing.” The wiki facilitated to share knowledge which did not efficiently share in traditional software development style. “Development know-how” in W15 of Appendix is one of the evidences for A2.

This development know-how was published by a member who had a similar development experiences. This member wrote politely the knowledge to treat with specific devices of such as tips of particular sensors or actuators control. This knowledge was based on the member’s own experiences and not formalized yet. In traditional software development, such know-how may be orally transferred among developers who joined the oral communication in TKN. In this case, the knowledge sharing in the wiki might avoid the rediscovery of this the knowledge to treat specific devices.

- Verifying A3: CMC tool can provide the appropriate communication occasions.

This assumption is related to “efficient interaction.” The wiki provide the appropriate communication occasions for software development. “Policy for test items (W7)” and “Comment for policy (W8)” of Appendix are the evidences for A3. In W7, a member published the policy for the test item for all the members in the wiki. In W8, Another member in other location replied for the policy by the wiki.

In traditional software development, this communication between members might suspend until regular weekly meeting. In this case, using the wiki provided the appropriate communication occasions and eliminated the delay factor in software development.

5 Discussions

We analyzed the case in former section. In this section, we discuss on the effectiveness and efficiency of wiki in the view points of “recording”, “effective knowledge sharing” and “efficient interaction.” We also discuss on the limitation of our analyses based on TIE model for software development communication.

5.1 Recording

We discuss the conditions that software developers record the important knowledge in CMC tools. We assume two reasons why they wrote the important knowledge in the wiki.

First reason is that the contents to be shared should be open. The wiki provided the developers with the open communication environment consistently. If they did not use the wiki, they might communicate by face to face meeting, telephone or e-mail. Wiki is more open than these communication methods. The open feature of wiki facilitates developers to write their knowledge. The open feature made casual

communication and the developers published the progress information each other. In face to face meeting, powerful members may interfere in the remarks of other low powered members. Wiki may facilitate the remarks of low powered members.

Second reason is that the content in wiki is a single object. In the case, they added the progress information to the document list. If they did not use the wiki, they might share the progress information of each document with e-mail. By e-mail, it is difficult to catch up with the progress information of all the members, because e-mail communication has the feature of cross in the post and makes multi objects to coordinate. The feature of cross in the post of e-mail caused distributes their knowledge. It is not efficient that someone should gather the distributed knowledge by e-mail. The members also need to read all the e-mail to comprehend the all the members' progresses. Because the shared contents feature is open and single object to edit, software developers record the important knowledge in CMC tools.

5.2 Effective knowledge sharing

We discuss the conditions that software developers can share the knowledge effectively. We assume that CMC usage is suitable for sharing developers' personal experiences. Particularly, developers' personal experiences are very useful knowledge for the software development with unfamiliar devices. If there is a member who treated with unfamiliar devices, the software development will advance smoothly with the developer's knowledge of unfamiliar devices. As these kinds of knowledge are not always systematized, the developers can share the knowledge with tacit knowledge network communication. CMC tools facilitate these kinds of knowledge to be share among the members. If this member wrote the knowledge in the wiki in 30 minutes and other eight members read the knowledge 5 minutes, the amount of the time is 70 minutes. On the other hands, if all the nine members acquired the knowledge by try and error in 120 minutes and the amount of the time is 1080 minutes. Although this is an extreme example, knowledge sharing in the CMC tools may be very effective. We assume that sharing unfamiliar knowledge is effective usage of CMC tools.

5.3 Efficient interaction

We now discuss the conditions that software developers can interact efficiently. We presume that two imaginary conditions of the software development communication; wiki style and traditional style. We suppose that wiki style has the regular weekly face to face meetings and wiki based communication. Traditional style is assumed to have only the regular weekly face to face meetings.

The relations between the amount of knowledge and time of each style can be described in Fig. 3 based on the above two conditions. Fig. 3 expresses the only the amount of knowledge increased by developers' communication, does not express the amount of the software development works. Straight line in Fig. 3 shows the increase of knowledge in wiki style. The dashed line also shows the increase of knowledge in traditional style.

The amount of increase of knowledge in wiki style is $Y(a,b)$.

$$Y(a,b) = a \cdot w_i + b \cdot m \quad (1)$$

In formula (1), a is the number of communication of wiki, w_i is the amount of knowledge increase per one wiki communication, b is the number of the face to face meetings and m is the amount of knowledge increase per one meeting.

The amount of increase of knowledge in traditional style is $Y(c)$.

$$Y(c) = c \cdot m \quad (2)$$

In formula (2), c is the number of the face to face meetings and m is the amount of knowledge increase per one meeting. To explain simply, w_i and m are fixed in this formula. However, w_i and m are variable by the features of communication of wiki and meetings in the real case.

We plot these two formulas in Fig.3 Both wiki and traditional styles have the first meeting in zero point in Fig.3. Both styles gain the same amount of knowledge by first meeting. In wiki style, developers may increase their knowledge to communicate each other three times via wiki. In traditional style, developers may not try to increase the knowledge to communicate each other. Although both styles increase the amount of knowledge by meeting, traditional style may gain the only half amount of the knowledge which wiki style may gain. As a result, wiki style will end the knowledge sharing at point EW_i , and its period L may be 3 weeks. Traditional style will end the knowledge sharing at point EM , and its period $L+D$ may be 6 weeks.

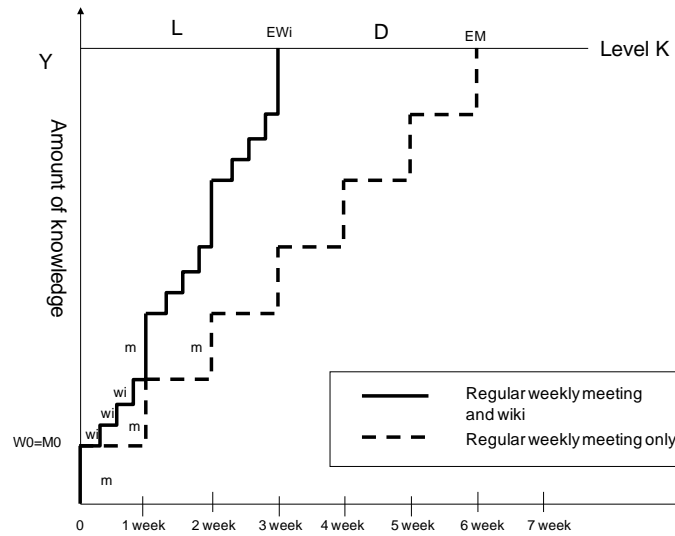


Fig. 3 The relation between time and amount of works

Fig. 3 is the suggestion that effective increases of communication occasions by the wiki cause the knowledge increase. In this suggestion, three interactions to gain the knowledge in wiki match one meeting communication to gain the knowledge. In traditional style, until the development team ends the knowledge sharing, they need seven meetings. In wiki style, they end the knowledge sharing with four meetings and

nine interactions in wiki. We suppose the wiki style may be more efficient knowledge communication environment than traditional style.

However, we also found a condition to establish our estimation in Fig.3. The condition is that the knowledge gained by wiki communication should be the same type of knowledge gained by face to face meetings. We supposed that in the real software development field, there are two types of knowledge. One is the knowledge which can be gained by CMC tools such as wiki. Another is the knowledge which can be gained only by face to face meetings. We should distinguish the knowledge to investigate the knowledge communication conditions under which CMC tools such as wiki work adequately.

5.4 Limitations

As analyzing the case, we discuss that CMC tools record the important knowledge and make software development process more effective and efficient under the certain conditions. We assure that CMC tools have the ability to facilitate the software development when the development team uses these tools well. In this article, we picked up the few positive effects of CMC tools. We should declare the conditions under which the CMC tools facilitate the software development more circumstantially. Also, we should analyze the negative factors of CMC tools such as false information in the wiki, information overload for developers. And we should investigate the relations among human, CMC content and document, which are the nodes of TIE model.

6 Summary

It is important to facilitate the knowledge communication among software developer. We proposed the intermediary knowledge model and TIE model for software development. Traditional software development researches focused on mainly human properties as experts and the quality of documentation. TIE model is software development communication model to explain the just in time documentation. We analyzed the case of the wiki based software development to show the effectiveness and efficiency of the wiki for software development. By analyzing the case, we discussed the conditions to facilitate the software development process by CMC tools. For further study, we should analyze more cases of CMC based software development by using our TIE model. We should also investigate the relation between the communication works and the other important works such as making document or coding.

References

1. Yamamoto, S and Kanbe, M. : Knowledge Creation by Enterprise SNS. The International Journal of Knowledge, Culture and Change Management. Vol.8 No.1 255--264 (2008)
2. Kanbe, M. and Yamamoto, S. : An Analysis of Computer Mediated Communication Patterns. The International Journal of Knowledge, Culture and Change Management. Vol.9 No.3 35--47 (2009)

3. Polanyi, M. : The Tacit Dimension, Routledge & Kegan Paul Ltd., (1966)
4. Nonaka, I. and Takeuchi, H. : The knowledge creating company How Japanese Companies Create the Dynamics of Innovation, Oxford Univ., (1995)
5. Rittel, H. and Kunz, W. : Issues as elements of information systems., Working paper# 1 31. Institute fur Grundlagen der Planung I.A.University of Stuttgart.
6. Conklin, J. and Begeman, M.L. : gIBIS: A Hypertext Tool for Exploratory Policy Discussion., ACM Transactions on Office Information Systems, 4, 6, pp. 303--331 (1988)
7. Conklin J., Selvin A., Shum S., B. and Sierhuis M. : Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS., Hypertext'01 Conference. (2001)
8. Ko, A.J. , DeLine, R. and Venoloa, G. : Information Needs in Collocated Software development teams., 29th International Conference on Software Engineering (ICSE'07) (2007).
9. Ye, Y., Yamamoto, Y. and Nakakoji, K. : Expanding the Knowing Capability of Software Developers through Knowledge Collaboration., International Journal of Technology, Policy and Management Vol. 8, No. 1, pp. 41--58 (2008)
10. Ye, Y., Yamamoto, Y., Nakakoji, K., Nishinaka, Y. and Asada, M. : Searching the Library and Asking the Peers: Learning to Use Java APIs on Demand., in V. Amaral, L. Veigaet al (eds.): Proceedings of 2007 International Conference on Principles and Practices of Programming in Java, ACM Press: Lisbon, Portugal, pp. 41--50 (2007).
11. Marczak, S., Damian, D., Stege, U. and Schroter, A. : Information Brokers in Requirement-Dependency Social Networks., 16th IEEE International Requirement Engineering Conference, pp. 53--62 (2008)

Appendix: Contents and intermediary knowledge transformation mode in the wiki in software development process

Names of pages	ID	Items	contents	Knowledge transformation mode
Basic design	W1	•Basic design documents List	•Member added the progress for each items.	Fragmentation, Publication
Detail design	W2	•Detail design documents List	•Member added the progress for each items.	Fragmentation, Publication
	W3	•Tasks	•Tasks in this process	Publication
Make and Unit test	W4	•FYI	•Discussion memo for the decision items	Publication
	W5	•Policy for test items	•Descriptions of policy for test items and conditions	Publication
	W6	•Estimation method of test density	•Estimation with number of test items and scales	Publication
System integration test	W7	•Policy for test items	•Descriptions of policy for test items and conditions	Publication
	W8	•Comment for Policy	•Comment for policy of W7	Collaboration
	W9	•Estimation method of test density	• Estimation with number of test items and scales	Publication
	W10	•List of the test materials	•List of the materials; software and hardware	Publication
Run time test	W11	•Call for comments	•Message to make the run time test guideline	Publication
Development conference #1	W12	•Development conference #1 document list	•Document list for Development meeting #1	Fragmentation
Development conference #2	W13	•Development conference #2 document list	•Document list for Development meeting #2	Fragmentation
Graph	W14	•the graph description specification	•Graph specification of system	Publication
Know-how	W15	•Development Know-how	•Know-how from the similar system experienced worker	Publication
Memo for Project management	W16	•Items to be improved for project management	•Communication method for project management	Publication
Demonstration	W17	•Purpose	•Purpose of demonstration	Publication
	W18	•Scenario	•Description of use cases for office and factory	Publication
	W19	•Proposal of the demonstration	•phases of demonstration and the To Do lists	Publication, Fragmentation, Collaboration
Name of Documents	W20	•Chapters	•Chapter and correction comments for documents	Fragmentation, Publication
	W21	•Documents list	•Documents list and working memos	Fragmentation, Publication