

Submission type: Research paper

# Hypermedia Support for Collaboration in Requirements Analysis

Kenji Takahashi<sup>1</sup>, Colin Potts<sup>2</sup>, Vinay Kumar, Kenji Ota<sup>1</sup>, and Jeffrey Smith<sup>1</sup>

<sup>1</sup>NTT Software Laboratories  
Nippon Telegraph and Telephone Corporation  
250 Cambridge Avenue, Suite 205  
Palo Alto, CA 94306 USA  
kt@nttlabs.com

<sup>2</sup>College of Computing  
Georgia Institute of Technology

## Abstract

We describe a model and system to support collaboration in requirements analysis with hypermedia technologies. The hypermedia support approaches the issues in communication, agreement, and traceability management, which are critical in the early phase in requirements analysis. Our approach consists of (1) an extension of the Inquiry Cycle model, a cyclic model of requirements analysis, for the collaborative work, and (2) comprehensive interaction media that fully utilizes WWW (World Wide Web) and Internet technologies. In this paper, we review the lessons learnt from experiences in applying the Inquiry Cycle model and Tuiqiao, a hypertext tool based on the model, to collaborative work in requirements analysis of a telephone directory service. We then propose an extension of the model and discuss the design of EColabor, a hypermedia system for requirements analysis. EColabor is based on the extended Inquiry Cycle model and implemented with the web and Internet technologies, including distributed version control and audio/video conferencing.

**Keywords:** collaboration, hypermedia, WWW, Internet, Inquiry Cycle, EColabor

## 1. Introduction

Recently, several field studies and experiments have been conducted to investigate work practice and problems in requirements analysis [Curtis88, Kaiya93, Lubars93, Takahashi95]. Results from these efforts commonly emphasize the importance of three activities: communication, agreement, and traceability management. However these activities are often done inefficiently and are poorly supported.

In response to these problems, we have developed the Inquiry Cycle model [Potts93a; Potts94], a cyclic model of requirements analysis, and a single-user support tool, Tuiqiao, based on the model [Takahashi94]. The model and tool have been applied to several projects [Potts94, Potts95a, Higuchi95]. In a previous paper, we describe our experience in using both in various types of in the requirements analysis of an Internet telephone directory service[Potts95a].

Our results reveal limitations in applying our method to collaborative situations and suggest some refinements: First, requirements analysis requires several styles of collaboration — from collaborative and synchronous to individual and asynchronous. Second, many decisions are postponed until the requirements are documented. These postponed decisions must not be forgotten, and categorized by the reasons for their postponement and the actions required of designers later. Finally, collaborative discussions have a more flexible structure and wider range of expression than those provided in the Inquiry Cycle model. Examples include new insights irrelevant to the “official” topic and transitory information expressed in spoken words or as informal sketches on a whiteboard.

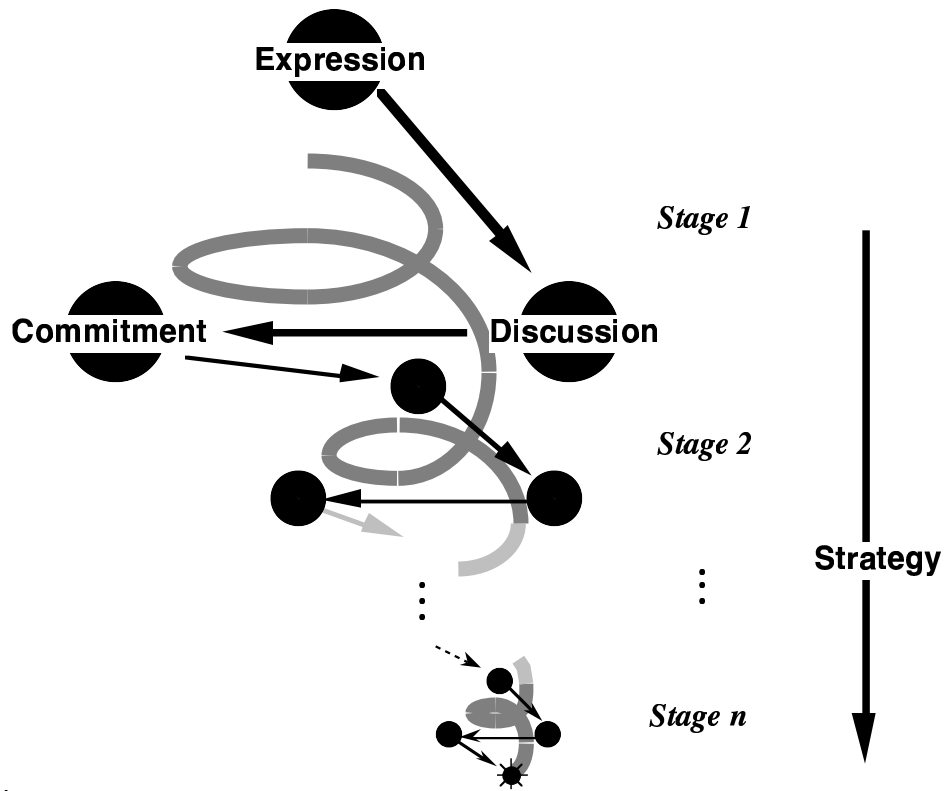
In this paper, we discuss our refined approach to supporting collaborative work in requirements analysis with hypermedia technology. Hypermedia technology can play an important role in helping a team keep track of various kinds of informal information that they generate and refer to (such as records of discussions, decisions made, agreements, and changes). First, we review the Inquiry Cycle model and the lessons learnt from applying it to real requirements projects. Next, we introduce our refinements to the model that are aimed at better support for collaboration through shared hypermedia documentation. We then describe EColabor, a group tool which is based on the refined model and which uses Internet and WWW (World Wide Web) technology for its implementation. We close by discussing the status of EColabor and our plans.

## 2. Inquiry Cycle

The Inquiry Cycle is a cyclic model for requirements analysis, which consists of three activities (Figure 1):

- (a) *Expression* is the proposing or preparing of requirements-related information, including not only requirements documents, but also domain-specific information, scenarios, and enterprise goals.
- (b) *Discussion* includes discussing requirements in formal meetings and circulating of comments and individual annotation of requirements.
- (c) *Commitment* includes making decisions based on the discussions, such as change requests, agreements about terminology, and commitments to seek missing information.

These three activities repeat under the control of a strategy to refine requirements until they become deliverable.



□Figure 1. The Inquiry Cycle

(The number and nature of the stages are controlled by the strategy)

The Inquiry Cycle is a generic model and can be instantiated to adapt to a specific organization, project, or purpose. Requirements analysis methodologies, such as scenario analysis and enterprise goal hierarchy analysis may be used during the three activities [Potts94, Potts95b]. For example, in developing the telephone directory system we instantiated the Inquiry Cycle with natural language text and the system's goal hierarchy for expression, a speech act theory [Searl69] that is similar to but simpler than IBIS (Issue Based Information System) [Kunz70] for discussion, and the goal hierarchy analysis method for strategy.

The Inquiry Cycle is an artifact-based model, which has two major advantages for collaborative work: (1) it makes changes traceable, and (2) it helps “participants<sup>1</sup>” in requirements sessions share awareness by making the artifact being discussed visible and explicit. “Awareness” is vital for effective collaborative work. For example, participants sometimes fail to understand what someone is talking about in a teleconference because they do not know what the speaker is looking at while speaking. Such problems rarely happen in face-to-face meetings because people know how to maintain awareness of focus.

Tuiqiao is a single-user hypertext tool for requirements analysis based on an instantiation of the Inquiry Cycle model [Takahashi94]. It allows users to keep track of textual requirements and their related information by linking them in a non-linear manner. The instantiation of the Inquiry Cycle model consists of textual descriptions and scenarios for expression, a simple speech act model that has only three categories (questions, answers, and reasons) for discussion, informal consensus leading to typed hypertext editing operations for commitment, and the scenario analysis method for strategy.

There are three possible views of the Inquiry Cycle model depending on how it is expected to contribute to requirements analysis; as a rhetoric for explaining one’s ideas and persuading others it contributes to effective communication and agreement; as a representation method of requirements analysis history it contributes to traceability; and as a process model it coordinates and guides participants toward an agreed specification.

---

<sup>1</sup> To avoid narrowing our discussion here too much, we use the neutral word “participant” for one who participates in a requirements analysis, such as analysts and customers.

### 3. Lessons Learnt

Here we summarize our experiences in applying the Inquiry Cycle model with Tuiqiao to the development of an Internet-based telephone directory service [Potts95a]. Tuiqiao is basically a single-user tool, but if used in Cogent, an advanced electronic meeting room [Kuwana95], it enables Tuiqiao and other single-user tools to behave as a collaborative tool. In the Cogent room, every participant has her/his own display and project her/his screen image to others. All the participants can thus share the same view of Tuiqiao — one of them just operates Tuiqiao on her/his display and projects the display image to the others. In the project, three sub-teams, who were located in different places (two in Tokyo and one in California), participated. They worked in various styles — synchronous collaboration with and without Cogent and Tuiqiao, asynchronous collaboration via e-mail, and individual work with Tuiqiao or an outline processor.

Tuiqiao is based on the instantiation of the Inquiry Cycle which assumes that (1) requirements documents to be discussed already exist before a requirements session, and (2) all the discussions correspond exactly to specific parts of the requirements drafts. Our experience with Tuiqiao, however, reveals limitations of these assumptions in a collaborative setting [Potts95]. For example, the subjects of discussions (artifacts) are created during the session as sketches on a whiteboard rather than prepared beforehand. Ongoing discussions include questions about questions, questions about answers, and so on. In addition, some discussions concern the entire target system, and completely new ideas are sometimes generated during such discussions. This made it difficult to determine which parts of the requirements these discussions should be linked to.

“Volatile” information was also important but could not be handled well by the current model and tools. Such information included a function list written on a whiteboard during a

brain-storming session, which was the first requirements draft, and spoken about design sketches of user interface drawn in a participant's notebook. Although this information rapidly evolved during discussions and was difficult to keep track of, it was a major driving force in the face-to-face meetings and contributed to the requirements documents. We need a mechanism to facilitate smooth transition from unstructured, volatile information to structured, solid documents.

Categorizing the status of discussions helped us manage requirements analysis process, because it suggested what actions to take next. For example, questions that required input from the customers (if recognized as being of this type) helped us to prepare for interviews with customers. This answers the common criticism of design tracking tools that they are not designed for the benefit of the people who actually use them (e.g., analysts) but for other people (e.g., maintainers) in the later development phases [Grudin88, Burgess-Yakemovic90].

Many work styles and tools were used during the requirements analysis. Our work styles shifted from synchronous collaboration to asynchronous and individual work as we progressed. Also our focus shifted from expression and discussion to commitment. For example, at first we regularly met and discussed a requirements draft, using Tuiqiao in the Cogent room. One participant then revised the document and distributed it to the others. We then discussed the document using e-mail. We could not easily move the results of one work style to another while maintaining traceability. We need a framework to integrate existing tools for synchronous, asynchronous and individual work.

#### **4. Hypermedia Approach**

We now describe an instantiation of the Inquiry Cycle model that seamlessly supports any combination of synchronous/asynchronous and distributed/collocated collaboration. Our instantiation (1) handles multimedia information, such as diagrams and video/audio records as well as text, (2) represents the requirements analysis more flexibly, including the status of the information, and (3) introduces a “reminder” type of discussion element that capture general and new ideas. Multimedia information captures artifacts in the process of creation during discussions while audio/video recording reduces the need to take notes.

## **4.1 Requirements Elaboration Strategy**

As the strategy for the instantiation, we use “goal decomposition” [Anton94] and “scenario analysis” [Potts94, Potts95b]. These are used to determine or elicit requirements and are orthogonal to the conventional methodologies, such as OOA/D and SA/SD, which presuppose prior analysis of the requirements. Although goal decomposition and scenario analysis can be systematized and the resulting methods learned like any other, elsewhere we show that analysts use both strategies naturally [Takahashi95]

## **4.2 Expression of Requirements**

Requirements are expressed in terms of *requirements documents*, *goal hierarchies*, *scenarios*, and *exhibits*.

### **Requirements document**

Requirements documents consist of two types of elements: textual descriptions and diagrams. Text descriptions are requirements written in natural language. Diagrams in

requirements documents typically have formal semantics, such as entity-relationship diagrams and dataflow diagrams, but informal illustrations can also be included.

### **Goal hierarchy**

The goal hierarchy represents what is to be achieved by the target system. In the hierarchy, goals are achieved by achieving subgoals. The goal decomposition method identifies goals and their subgoals until they are specific enough to be operationalized.

### **Scenario**

A scenario is one or more (usually several) end-to-end transactions involving the required system and its environment. There are three kinds of scenarios: generic scenarios, specialized scenarios, and episodes. A generic scenario is a sequence of actions, each consisting of three components: a description, participating object classes, and services involved in the action. A generic scenario can be specialized into several specialized scenarios, each of which corresponds to a different situation. Each element of a generic scenario is accordingly specialized into a detailed description, a set of objects, or a set of services. Episodes are the building blocks of generic scenarios. Episodes are ‘phases’ of activity and are therefore scenario fragments. Each episode illustrates the achievement or obstruction of a goal.

### **Exhibit**

Exhibits capture the “volatile” ideas generated during collaboration. Exhibits include spoken comments and sketches drawn on whiteboards. Eventually, useful exhibits are transcribed and incorporated into the requirements documents.

Exhibits are expressed in four different types of media: audio, video, sketches, and text.

Audio and video capture ideas and the process of their being presented. Written ideas are

represented in sketches and text. All pieces of such information are synchronized and interconnected. For example, consider a sequence of actions in which participants draw, discuss, and modify a sketch. An audio/video record of the discussion links the original to the modified sketch so that the sequence can be easily traced.

Sometimes no requirements document is available, particularly at initial meetings. These sessions produce a requirements expression that consists only of exhibits.

### **Version**

A version of the expression represents requirements agreed upon at a certain point. An expression consists of requirements documents, exhibits, goals, and scenarios. The overall expression and each element of the document have different and independent versioning schemes.

This distinction between versioning schemes of entire documents and their elements allows us:

- (1) to work on specific elements in parallel while maintaining consistency, and
- (2) to create a new requirements document for a variant system by using elements of any versions as building blocks.

For example, imagine version 1 of an expression (EX<sub>1</sub>) consisting of two paragraphs of a requirements document (P1<sub>1</sub> and P2<sub>1</sub>) and a description of the goal hierarchy (GH<sub>1</sub>). After a discussion of EX<sub>1</sub>, P1<sub>1</sub> is revised to P1<sub>2</sub>. Then, the participants decide to make version 2 (EX<sub>2</sub>) of the expression from P1<sub>2</sub>, P2<sub>1</sub>, and GH<sub>1</sub>. They also produce a variant (P1<sub>a1</sub>), that is, version 1 of variant 'a' (called version 'a1') of P1<sub>1</sub>, and define version 'a1' (EX<sub>a1</sub>), which consists of P1<sub>a1</sub>, P1<sub>2</sub>, and GH<sub>1</sub>. The transitions between these versions are illustrated in Figure 2.

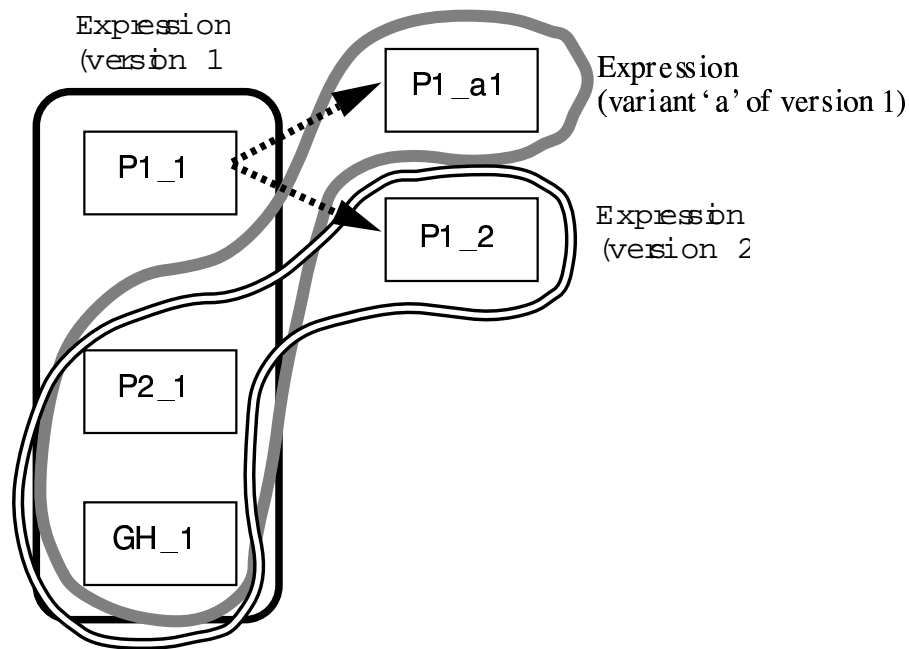


Figure 2. Transitions between versions of a requirements expression

### Interrelationships within expressions

The elements of a version of an expression are interrelated by three types of relationships: “previous/next”, “subsumes”, and “related to”. The “previous/next” and “subsumes” relationships are used to represent the expression in a linear way like a printed document: “previous” and “next” determine the sequence of the expression, and “subsumes” defines which element includes others. For example, an element of the expression that corresponds to a chapter subsumes other elements that correspond to sections. “Related to” links related elements of the expression — for example, an element and its references.

There are also “old/new” relationships between old and new versions of elements of the expression. Such relationships can be established between versions of the expression.

## 4.3 Discussion of Requirements

Discussions are represented by two types of elements: inquiry and reminder. Inquiry elements — questions, answers, and reasons — form the main thread of a discussion. These elements are essential and fixed in this Inquiry Cycle instantiation. Reminders are elements to capture “free floating” discussions, which are not directly related to a particular part of the expression. These two types of elements can be specialized and/or decomposed further according to a specific decision-making policy or procedure that participants follow. For example, some participants apply a more formal policy that specifies and manages the deadline for each task. They need to further categorize the reminder type, such as scheduling and responsibility assignment, to keep track of the deadlines.

### **Inquiry elements**

The three types of inquiry elements (questions, answers and reasons) are interconnected and each one has a status (e.g., resolved or unresolved).

### **Interrelationships between inquiry elements**

In the instantiation of the Inquiry Cycle model for the single user tool, Tuiqiao, only the requirements can be the subject of a discussion. However, in the collaborative work instantiation, any element of the hypermedia record, including exhibits, inquiry elements, and even change requests (see “Commitment” below), can be the subject of a discussion, so the elements can be flexibly interconnected. For example, questions are interconnected for various reasons:

*Dependency:* Some questions cannot be answered until other questions have been answered.

*Clarification:* Questions may be asked to clarify the meaning of other questions. This usually happens in asynchronous sessions. In synchronous sessions, these issues are

immediately resolved within the session, so the clarification questions do not need to be recorded.

*Decomposition:* When a question covers a wide scope, analysts decompose it into several sub-questions. The original question often cannot be answered until all the sub-questions have been answered.

### **Status of inquiry elements**

Inquiry elements can be resolved or unresolved. This categorization indicates the progress of the analysis process and helps us track responsibilities.

We further break down unresolved element, according to the reasons they remain unresolved. Doing so helps us know what actions to take to resolve them (Table 1). We have identified five reasons for unresolved discussion elements:

- deferred until implementation
- waiting for customer input
- needing additional work (such as a technical feasibility study, market research, performance analysis)
- depending on others (i.e. the resolution of other elements), and
- other (e.g., pending discussion)

Table 1. Status and actions to take

Values of status	Actions to take
Deferred until implementation	Hold status until implementation, and then take into consideration.
Waiting for customer input	Compile the elements as the interview agenda for next session with the customers.
Needing additional work	Schedule and conduct additional work, and address the elements based on the results.
Depending on others	Maintain status until the other elements have been resolved, and then address these elements accordingly.
Other	Prioritize and schedule; these elements may turn out to have a different status after discussions.

## Reminder

Some elements cannot be related directly to a specific element of the requirements expression. We call it “reminder”. The reminder type is a “parts bin” of useful ideas [Constantine93]), including:

- implementation directives or warnings, and
- project management reminders, such as assigning responsibilities, checking the status of projects, and scheduling.

## 4.4 Commitment

There are six types of commitments: refinement, clarification, retraction, merge, split, and transcription. *Refinement* results in more complex structural alternatives: one or more requirements are derived from the refined requirement. *Clarification* results in the rewording of one or more requirements. Retraction deletes a requirement. *Merging* generates a new requirement from several requirements. *Splitting* generates several requirements from one requirement. *Transcription* transcribes and incorporates informal exhibits into a new version of an expression. These commitments are shown in Figure 3.

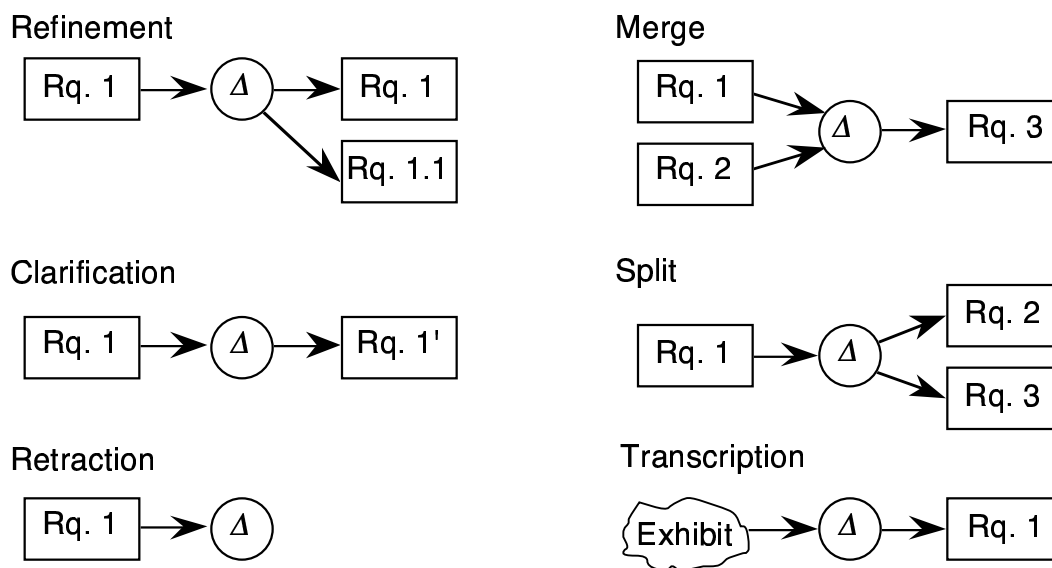


Figure 3. Types of commitment

## 5. EColabor: A Hypermedia System for Requirements

### Analysis

EColabor is an active hypermedia system for collaboration in requirements analysis based on the Inquiry Cycle model described in Section 4. EColabor manages as networked hypermedia the pieces of information generated and referred to during requirements analysis — such as requirements documents, video records of discussions about the documents, and change histories. For better communication and faster agreement, it

transmits the information in various media and modes, including audio/video conferencing. For ensured traceability, it keeps track of the evolving requirements documents along with the informal background information. For efficient coordination, it monitors and navigates the requirements analysis process, including sending notifications at particular events.

We decided to implement EColabor using Internet and WWW technologies, because of their interoperability and large user populations of these technologies potentially solve the “incompatibility” and “critical mass” problems that are common to collaborative tools [Markus90]

## **5.1 Architecture**

EColabor has a client-server architecture that provides participants, even if distributed or working asynchronously, with transparent access to the EColabor hypermedia information. The client-server architecture also allows each participant to shift seamlessly between her/his private working space and the shared public working space for a requirements analysis project, but with well-maintained integrity and well-protected privacy. The architecture of EColabor is shown in Figure 4.

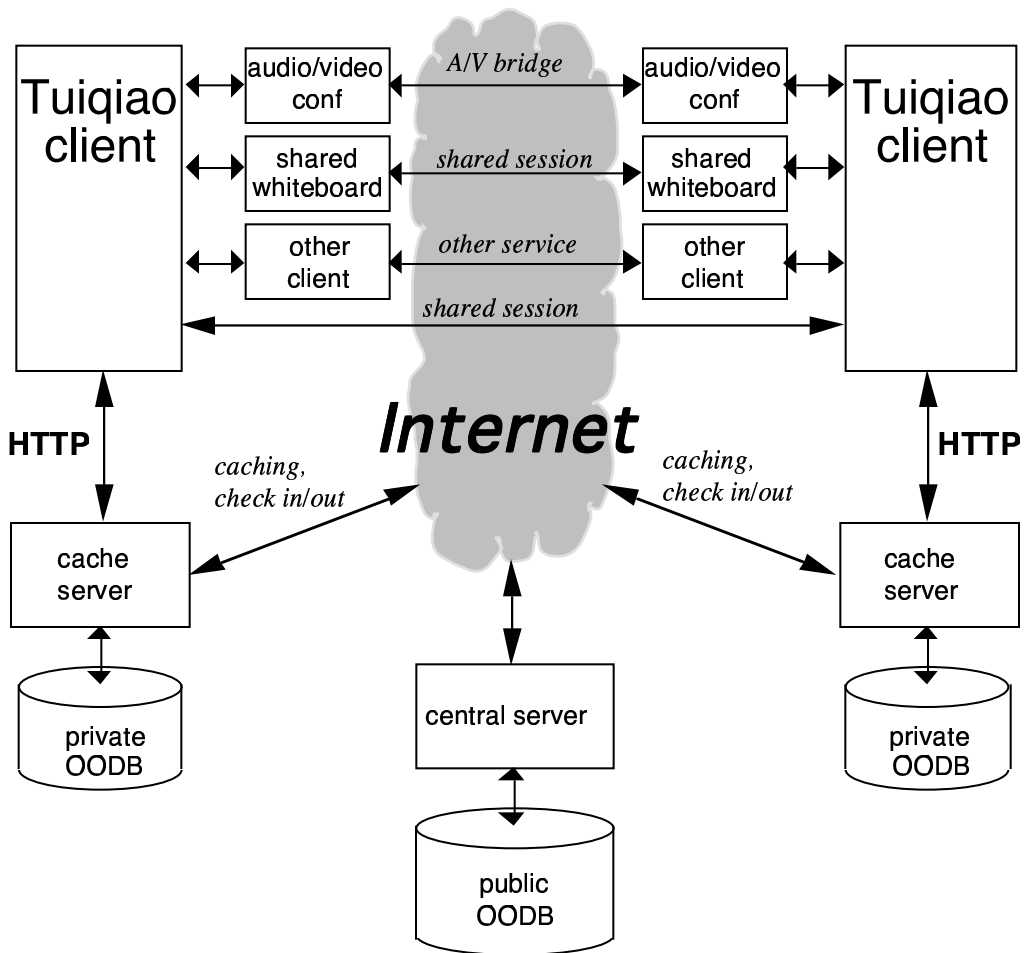


Figure 4. Client-server architecture of EColabor

The EColabor hypermedia information is stored and maintained in a public OODB (object-oriented database). Participants access this public database through cache servers and some data is cached in their private databases. Participants can store private documents and notes in their private databases. They can also check out the public data to the private databases and check in revisions. The central server is responsible for managing the public database, including access control and version control. Tuiqiao clients communicate with video/audio conferencing tools, shared whiteboards, and other clients. They also handle

interactions with users. Other clients, such as e-mail clients, can be connected to the Tuiqiao clients via an API (application programmer's interface).

Let's consider two examples to see how this architecture works:

(1) Distributed participants can have a synchronous requirements meeting by using video/audio conferencing, Tuiqiao clients and shared whiteboards via LAN/WAN. The Tuiqiao client's connections with the teleconferencing tools and the shared whiteboards enable the video/audio streams and the images on the whiteboards to be recorded, indexed, and linked to relevant elements of the EColabor hypermedia information during the meetings.

(2) Asynchronous participants can also access elements of the information to review records of discussions about it, replay video segments related to it, and so on. If the participants have comments on the element, they can submit the comments to the public database via e-mail. The participants can also make changes to parts of requirements documents in parallel and then submit them into the public database by explicitly checking them in or out.

Basic mechanisms include:

- (1) assigning an URL to each element of the hypermedia information stored in the OODB,
- (2) extending HTTP (hypertext transfer protocol) to handle check in/out of versions of documents and using the extension for communications between the Tuiqiao clients and the servers,
- (3) multicasting HTTP responses via MBone (multicast backbone) [Macedonia94], a virtual network for multicasting multimedia data, to share the views of and the information handled by Tuiqiao clients, and
- (4) multicasting the audio/video streams via MBone for audio/video conferencing.

## 5.2 Communication support

EColabor provides the following communication support:

- multi-point audio/video conferencing
- semi-structured e-mail
- shared electronic whiteboard
- application sharing

A suite of conferencing tools, including mmphone, sharedmosaic, and Xshare [Kumar94], is available to users both on the desktop computers in their offices as well as in Cogent. Cogent allows participants to seamlessly maneuver views between private to shared space via electronic and traditional presentation tools. Streams of the audio/video conferencing are automatically segmented and captured by traceability management functions (see 5.3 Traceability support).

Semi-structured e-mail is used in asynchronous sessions. The e-mail messages are categorized into four types, each corresponding to a different inquiry element type: question, answer, reason, or commitment. The messages can be represented in text, pictures, audio and/or video. We use Storyboard, a MIME (multiple internet mail extensions) [Borenstein93] multimedia e-mail tool [Kumar94], to compose, send, receive, and replay such multimedia messages in a WYSIWYG manner. EColabor then puts the messages into the appropriate places in the hypermedia information according to the senders' prescription in the messages.

### **5.3 Traceability Support**

EColabor lets participants keep track of evolving process of the requirements documents in multimedia. According to the Inquiry Cycle model, pieces of the documents, questions, answers, reasons, and commitments are recorded in multimedia and interconnected.

#### **Selection triggered segmentation of audio/video record**

EColabor continuously records an entire session in audio/video and provides support for segmenting the records on the fly, using the “selection triggered segmentation” method similar to Synthesis, a video-based collaborative writing tool [Potts93b].

The method asks participants to select the elements of the EColabor hypermedia information that they are discussing as specifically as possible on their computer displays. These selections are used to let the other participants and EColabor itself know what is currently being discussed. This ensures that “awareness” is shared among the participants and also enables EColabor to segment the audio/video records and link them to the related pieces of the information. Triggered by the selections, EColabor segments and records the audio/video streams, and links the segments to the selected pieces (that is, the ones being discussed). Newly created elements are selected by default. For example, EColabor attaches a “beginning” tag to the current point of the video/audio stream when a participant selects and starts discussing a paragraph of the requirements document in the public space, and attaches an “end” tag to the current point when the participant selects another paragraph. The audio/video segments may overlap.

The participants can also explicitly segment the audio/video streams and label the segments. They segment and label a topic being discussed in a synchronous session. The labeled

topic is a temporal record — it is not categorized according to the Inquiry Cycle model because it may contain several types of the elements. After the session, the participants should transcribe appropriate element types from the audio/video record of the topic.

EColabor supports taking snapshots of sketches and text on the shared whiteboards. Segmentation of the audio/video records is triggered when a snapshot is taken on the assumption that participants come to an agreement at that point. The audio/video segment between the current and previous snapshots explains how and why the current one is produced, and thus is linked to the current one. The elements being selected during this period are linked to both the segment and the snapshots.

## **5.4 Coordination support**

### **Status Management**

EColabor identifies and reports the status of the requirements analysis process. EColabor allows users to define values of the status of the discussion elements in the hypermedia information and to set any of the values (see Status in 4.3 Discussion of Requirements). To display status, EColabor uses styles and colors in the Tuiqiao client and provides functions for status retrieval. A listing of the current status can be produced at any time.

More importantly, EColabor can keep progress visible and prompts for or executes required actions. Participants can specify these actions - what to do and when (for example, sending a customer an e-mail message asking a question when its status is set to “ask the customer”).

## Version Control

EColabor lets participants control access to the working version, and define a new version. It manages the status of discussions over revision.

We use the "baseline" approach to version control, in which an official baseline version is shared by all participants. Each element can be checked out, modified, and then checked in to be included in a new baseline version. A variant of the baseline can be created for another project to develop a variant system.

Check-in and check-out processes may be done in parallel, and thus several participants can be working on the same expression simultaneously. When checked in, each element is uniquely and automatically named as a new version by the central server. The access right to read and/or write an element can be defined and is managed by the central server. When all participants have finished editing, they define a new baseline version of the expression and check in it to the central database. The expression version is also uniquely and automatically numbered by the central server.

The status of elements of the baseline version is recorded and maintained in a version control table, which is managed by the EColabor server. The version control table consists of the following columns:

- *Requirements* indicates elements of the requirements document.
- *Who is working on* monitors who is working on each element.
- *Permission* controls who has read/write right to each element.
- *Action* specifies what should be done and when.
- *Current working variant* keeps track of working variant of each element that is latest but not checked in as an element of the baseline version. □

For example, in Table 2, three participants, Jeff, Ota, and Kenji, are involved in a concurrent editing session. They are independently working on a requirements expression, which consists of Rq1, Rq2, and Rq3. Jeff has checked out Rq1 and checked in Rq4 as a new version of Rq1. Kenji has also checked out Rq1 and checked in Rq5 and Rq6 as variants of Rq1. Rq3 has also been checked out and changed into a new version, Rq7, by Kenji. According to the "Action" fields, a mail message is sent to all participants when Rq1 is checked in and when Rq3 is checked out. Rq1 can be read and edited by all the three participants, Rq2 by Jeff and Ota, and Rq3 by Ota and Kenji.

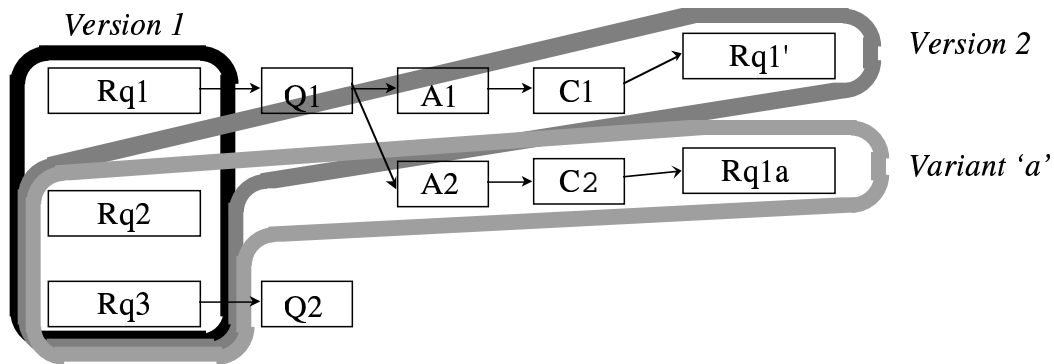
Table 2. An example of version control table

Requirements	Who is working on	Permission	Action	Current working variant
Rq1	Jeff Kenji	Kenji(rw) Ota(rw) Jeff(rw)	Send mail to all when checked in	Rq4 (by Jeff) Rq5, Rq6 (by Kenji)
Rq2		Jeff (rw) Ota(rw)	Send mail to Kenji whenever	
Rq3	Kenji	Ota(rw) Kenji(rw)	Send mail to all when checked out	Rq7 (by Kenji)

The participants can even go back to an earlier version and produce a variant that branches at a certain point in the discussions on the version. For example, a variant is produced by selecting another alternative answer that was not taken as follows(Figure 5). Version 1 of requirements document, which consists of three elements (Rq1, Rq2, and Rq3), is being analyzed. The first, Rq1 is challenged by a question (Q1), two alternative answers (A1 and A2) are answered to Q1 (a question (Q2) is also raised but not answered). Next, A1 is selected and a change request (C1) is suggested based on A1. Then according to C1, Rq1

is changed into Rq1'. Finally the version 2 of the document, which consists of Rq1', Rq2, and Rq3, is defined. Afterwards, someone traces back through this analysis process, select the other answer (A2) and thinks of a change request (C2). According to C2, Rq1 is changed into Rq1a. Then s/he defines a variant of the document, which consists of Rq1a, Rq2, and Rq3.

Some discussions and commitments remain postponed even after a new version has been created. EColabor manages the status of the requirements analysis process over versions and displays the status accordingly. In the example above, because Q1 is resolved in the process of revising version 1 into version 2, Q1 should not be displayed along with the version two. Q2, however, should be displayed because it still remains open (Figure 5).



□Figure 5. Variant of documentation

Based on the version control table, EColabor provides the following four functions for controlling versions:

*Access right management* lets participants set the value of the "Permission" column of the version control table in order to qualify or disqualify a participant to read or edit an element of the expression.

*Status monitoring* monitors and records who is working on which element in the "Who is working on" column of the table.

*Coordination* helps participants coordinate their editing processes based on the scripts in the "Action" column. For example, a participant prepares a script that sends a notification to participants who are concurrently editing the same element of an expression when any modification is made to that element to avoid inconsistency in their editing results.

*Merging support* composes a suggested version of the current working variants upon participants' requests based on the "Current working variant" column. For example, based on Table 2, EColabor suggests a new version, which consists of Rq4 (or Rq5 and Rq6), Rq2, and Rq7.

## **5.5 Status**

EColabor is being implemented in three globally distributed sites (Tokyo, Palo Alto and Atlanta). Most of the components of EColabor, including the Tuiqiao client and multimedia communication tools, exist. Our development efforts are focused on the server that integrates the components. We are applying the Inquiry Cycle and the prototype to our own development process.

## **6. Discussion**

Several researches have addressed communication, agreement, and traceability during requirements analysis. None of them, however, have integrated these three subjects. For

communication and agreement, single user tools (e.g., gIBIS [Conklin89]) exist as well as collaborative tools (e.g., SIBYL [Lee90]). These tools, however, do not relate the supported communications to the artifacts which they are about and contribute to. Therefore they fail to ensure the traceability.

On the other hand, tools for traceability management do not sufficiently support communication and agreement in requirements analysis [Gotel94]. To manage documents and programs, many configuration management systems have been developed and practically used. Because these systems assume that the documents have a certain structure and/or are being produced following a certain procedure [Humphrey89], it is difficult for the systems to support the early phase of the requirements analysis process that involves intensive interactions among the participants and rapid changes of the informal and unstructured documents.

The parts of our approach are not new. Rather our focus is on:

- (1) a comprehensive model to integrate these parts for seamlessly supporting collaboration in requirements analysis, and
- (2) hypermedia technologies that implement the integration by fully utilizing the Internet/WWW.

Future work includes full implementation of EColabor, field studies in global collaboration with EColabor, and incorporation of EColabor into a collaboration environment (e.g., Cogent). We will also promote the standardization of the extension of HTTP and other related specifications for collaborative work via the web.

## References

- [Anton94] Anton, A., W.M. McCracken and C. Potts “Goal Decomposition and Scenario Analysis in Business Process Engineering”, *Advanced Information Systems Engineering, Proc. 6th Int. Conf. CAiSE'94*, Utrecht, Springer-Verlag Lecture Notes in Computer Science, 811, 1994.
- [Borenstein93] Borenstein, N., “MIME (Multiple Internet Mail Extensions) Part One: Mechanism for Specifying and Describing the Format of Internet Message Bodies”, *Internet Network Working Group Request for Comments, RFC-1251*, 1993.
- [Burgess-Yakemovic90] Burgess-Yakemovic, K. C. and J. Conklin, “Report on a Development Project Use of an Issue-Based Information System”, *Proc. ACM 1990 Conf. Computer-Supported Cooperative Work (CSCW'90)*, pp. 105-118, 1990.
- [Conklin89] Conklin, J. and M. Begeman, “gIBIS: A Tool for all Reasons”, *J. Am. Soc. Inf. Sci.*, May 1989, pp. 200-213.
- [Curtis88] Curtis, B., H. Krasner and N. Iscoe, “A Field Study of the Software Design Process for Large Teams”, *Comm. ACM*, Vol. 31, No. 11, pp.1268-1287, 1988.
- [Gotel94] Gotel, O. C. Z and C. W. Finkelstein, “An Analysis of the Requirements Traceability Problem”, *Proc. Int. Conf. Requirements Eng.*, pp. 94-101, 1994.
- [Higuchi95] Higuchi, M. and Takahashi, K., “World Wide Collaborative Writing: A Case Study”, *Proc. Conf. on Asian-Pacific World Wide Web*, to appear.
- [Humphrey89] Humphrey, W.S., *Managing the Software Process*, Addison Wesley Publishing, 1989

- [Kaiya93] H. Kaiya and M. Saeki, "A Supporting Tool for Face-to-face Meetings to Develop Software Specifications" (in Japanese), IEICE Technical Report, KBSE93-13, pp.9-16, 1993.
- [Kumar94] Kumar, V., and J. Glicksman, "A SHARED Web To Support Design Teams", Proc. IEEE Third Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 178-182, 1994.
- [Kunz70] Kunz, W. and H. Rittel, "Issues as Elements of Information Systems", Working Paper 131, Inst. Urban and Regional Development, Univ. California at Berkeley, 1970.
- [Kuwana95] Kuwana, E., E. Yana, Y. Sakamoto, Y. Nakamura, and K. Horikawa "Computer-Supported Meeting Environment for Collaborative Software Development", *Information and Software Technology*, Butterworth Hinemann UK, to appear.
- [Lee90] Lee, J. "SIBYL: A Tool for Managing Group Design", Proc. CSCW'90, pp. 79-92, 1990.
- [Lubars93] Lubars, M., C. Potts, and C. Richter, "A Review of the State of Practice in Requirements Modeling", Proc. RE'93, pp. 2-14, 1993.
- [Macedonia94] Macedonia, M. R., and D. P. Brutzman, "MBone Provides Audio and Video Across the Internet," IEEE Computer, Vol. 27, No. 4, pp. 30-36, 1994.
- [Markus90] Markus, M.L. and Connolly, T., "Why CSCW applications fail: Problem in the adaption of interdependent work tools", Proc. CSCW'90, pp. 371-380, 1990.
- [Potts93a] Potts, C. and K. Takahashi, "An Active Hypertext for System requirements", Proc. 7th Int. Workshop on Software Specification and Design, pp. 62-68, 1993.

- [Potts93b] Potts, C., J. D. Bolter, and A. Badre, "Collaborative Pre-Writing with a Video-Based Group Working Memory, Graphics Visualization and Usability Center Technical Report 93-35, Georgia Institute of Technology, 1993.
- [Potts94] Potts, C., K. Takahashi and A. I. Anton, "Inquiry-based Scenario Analysis of System Requirements", IEEE Software, Vol.11, No. 2, pp.21-32, 1994.
- [Potts95a] Potts, C., K. Takahashi, J. D. Smith, and K. Ota, "An Evaluation of Inquiry-Based Requirements Analysis for an Internet Service", Proc. RE'95, pp. 172-180, 1995.
- [Potts95b] Potts, C., "Using Schematic Scenarios to Understand User Needs", Proc. Symposium on Designing Interactive Systems (DIS'95), to appear.
- [Searle69], Searle, J. R., *Speech Acts*, Cambridge University Press, 1969.
- [Takahashi94] Takahashi, K. and C. Potts, "Tuiqiao: A Hypertext Tool for Requirements Analysis", Tech. Report GIT-CC-94107, Georgia Institute of Technology, 1994.
- [Takahashi95] Takahashi, K. and S. Yamamoto, "An Analysis of Traceability in Requirements Documents", IEICE Transactions on Information and Systems, Vol. E78-D, No.4, pp.394-402, 1994.