Mejoras de coordinación de conversaciones electrónicas en proyectos distribuidos de ingeniería de software

Inés Pederiva

20 de mayo de 2005

Índice general

Índice de figuras

Capítulo 1

Introducción

La comunicación entre personas dentro de un grupo de trabajo es una actividad muy compleja ya que informar y mantener comunicado a un grupo de personas puede fracasar si no está bien organizada la forma de comunicación, en el sentido de saber quienes dentro del grupo manejan qué información y a quienes deben ser los receptores de qué información. Esto ocurre tanto en la interacción de los grupos pequeños como de los extensos. Esta tesis analiza deficiencias en las comunicaciones que pueden encontrarse dentro de grupos de desarrollo de software cuya comunicación se realiza a través de las tecnologías existentes basadas en computadoras y propone una solución basada en esas tecnologías.

La diversidad de herramientas tecnológicas que surgen a partir de la existencia de internet permiten soportar diferentes formas de comunicación. Es posible entonces poder contar con herramientas para comunicarse en forma escrita, tanto sincrónica como asincrónicamente, o contar con comunicaciones de voz y / o video entre otras. A pesar de las nuevas posibilidades que surgen con estas tecnologías, los problemas en las comunicaciones dentro de los grupos de trabajo siguen existiendo.

Pensemos el caso de un proyecto con un cliente en España que necesita un producto y un grupo de trabajo para la implementación del mismo en Argentina. Existen dos características fundamentales en este tipo de situaciones. La primera es la diferencia horaria, ya que la existencia de un desfasaje en el comienzo del día laboral entre las partes puede afectar a que alguna de ellas no esté disponible en el momento en que la otra parte lo requiera. Este tipo de problemas puede ser evitable, porque con 4 o 5 horas de diferencia se puede establecer, dentro de los horarios laborables de las partes, una franja horaria en común que resuelva este problema. La segunda característica es la distancia entre el cliente y quien desarrolla el proyecto. En la actualidad

los problemas de comunicación derivados por la distancia pueden ser notablemente reducidos, ya que se puede contar con distintas formas de soporte groupware [1] para la comunicación que la lleven a ser inexistente. Si bien la comunicación presencial no puede resolverse, puede usarse una forma de comunicación sincrónica; como el teléfono o herramientas soportadas por Internet como el Chat, el audio o el video y audio; o una forma de comunicación asincrónica como por ejemplo el mail.

El proyecto mencionado existió realmente, con las características antes mencionadas y ante la disponibilidad de Internet de las partes, se definió que la forma de comunicación sería por algún medio relacionado con Internet. Así, la información del contacto de uno de los miembros del cliente fue difundida en el grupo de desarrollo en Argentina y se estableció la comunicación vía Chat y / o vía correo electrónico. Las comunicaciones entre las partes estaban establecidas y se consultaban una con otra por estos dos medios.

Pero los problemas de comunicación comenzaron a surgir hasta que las mismas resultaron incontrolables debido a que cualquier miembro tenía la posibilidad de hablar con el cliente y no había registro de los temas tratados ni de las decisiones tomadas. Fue entonces que dentro del grupo de trabajo en Argentina comenzaron a surgir conflictos ya que las decisiones tomadas entre algunos miembros y el cliente no eran debidamente comunicadas a todos quienes estaban afectados o interesados por las mismas, además de que el mismo líder del proyecto estaba desinformado sobre los temas tratados con el cliente.

Al pasar los días, lo que inicialmente era una diferencia menor sobre la interpretación de algo conversado con el cliente resultó ser un conflicto significativo en la etapa de integración del sistema. Surgieron en ese momento, diferencias de diseño que afectaban al producto final y el adaptar las partes para lograr la integración condujo a una pérdida significativa de tiempo. Como consecuencia la planificación se vio afectada por estos cambios y los tiempos se vieron significativamente alterados.

Las diferencias en las interpretaciones de las directivas del cliente y la falta de formalización que tenían las conversaciones hacían que no solo se encuentren inconsistencias en las interpretaciones, sino que tampoco había un lugar donde encontrar lo que el cliente realmente pedía. Esta falta de formalización existía por mantener las conversaciones a través de distintas herramientas de chat, correo electrónico y hasta por teléfono. Diferencias en las directivas recibidas por el cliente se encontraban cada vez con mas regularidad y era común que ante la posibilidad de preguntarle al cliente

[1] Groupware: sofware para trabajo en grupo.

otra vez sobre el comportamiento de cierta funcionalidad se volviera a recibir una directiva distinta a las anteriores.

Debido a que esta condición fue detectada avanzado el proyecto, no se pudo revertir la situación y los historiales en los chats, junto con los correos electrónicos enviados, terminaron formando parte de la documentación entregada cuando se entregó formalmente el producto final.

Se puede ver que la comunicación es mas compleja que la simple posibilidad de que todos los miembros de un equipo estén conectados. El hecho de que la gente esté conectada no implica que esté comunicada y mucho menos que esté "bien comunicada". Tampoco implica que las cosas importantes lleguen en tiempo y forma a quienes debe llegar. Inicialmente puede pensarse que los problemas se debían a la distancia con el cliente y la imposibilidad de tratar personalmente temas importantes referidos al proyecto. Pero los problemas detectados se repitieron en otros proyectos, inclusive en los de desarrollo local. Esto llevó a analizar detenidamente la comunicación en los proyectos de software.

La necesidad de planificar las comunicaciones va mas allá de una simple conexión ya que se debe analizar qué circuito deben tomar las modificaciones o indicaciones que surjan para que todos los involucrados en los cambios estén notificados de los mismos.

Sobre la base de intentar solucionar los problemas de comunicación y teniendo presente que existen problemas que van mas allá del alcance que se puede dar en un trabajo de grado en informática, esta tesis plantea problemas detectados y alternativas de soluciones para lograr organizar las comunicaciones y brindar métodos para lograr comunicar grupos de trabajos, tanto a nivel interno como a niveles ínter grupos dentro de una organización a la vez que también en el trato con el cliente.

Las contribuciones de esta tesis son:

- El análisis de los problemas de comunicación en proyectos de desarrollo de software distribuido.
- La propuesta de un modelo para organizar la comunicación en grupos de desarrollo de software distribuido basado en el concepto de conversaciones.
- Una implementación prototípica del modelo en la forma de una herramienta groupware de cuatro capas, basado en un servidor de mail extendido

La estructura de este trabajo está organizada en los siguientes capítulos.

En el Capítulo 2 se presenta el escenario sobre el cual se diseña el modelo propuesto. Se analizan todos los problemas de comunicación que surgen en un proyecto de software distribuido y se presentan los requerimientos que el modelo a proponer debe contemplar.

En el Capítulo 3 se describen las tecnologías existentes que dan soporte a las comunicaciones basadas en computadoras. Además, se presentan los factores que determinan las deficiencias que estas herramientas tienen para los problemas planteados.

Las primeras soluciones encontradas al problema de la coordinación de conversaciones que fueron aplicadas a proyectos reales, son presentadas en el Capítulo 4. También se describen las fallas por las cuales fracasaron estas soluciones tempranas.

El diseño de una herramienta groupware en términos genéricos se presenta en el Capítulo 5, haciendo una descripción del modelo de objetos y de la interfaz de la aplicación.

En el Capítulo 6 se presenta una implementación prototípica con tecnologías concretas sugeridas para el desarrollo final del modelo propuesto.

Por último en el Capítulo 7 se presentan las conclusiones y trabajos futuros.

Capítulo 2

Análisis del problema

En este capítulo se presenta el escenario que motiva al trabajo realizado. Presenta también los requerimientos a ser considerados en el diseño de un modelo que solucione los problemas de comunicación que surgen en los proyectos distribuidos de ingeniería de software.

2.1 Escenario

Como introducción se presentó un problema real ocurrido en un grupo de trabajo en Argentina, que desarrollaba un producto de software para un cliente en España. Este cliente, no era el usuario final del producto, sino una empresa de desarrollo que había tercerizado en Argentina un trabajo que le estaba haciendo a su cliente español. La forma de comunicación entre los dos grupos se estableció a través de Internet mediante el uso de Chat y el correo electrónico. En particular, para la comunicación por chat se usó el ICQ [17],

En un proyecto como el que se relata, entre Argentina y España, no surgen problemas de idioma y por ende se descartan los problemas que pueden surgir en este aspecto ya que ninguna de las partes necesita hablar un idioma no nativo para comunicarse. Además, como las partes que se comunicaban a la distancia eran ambas técnicas, no surgieron conflicto de jergas o términos técnicos que complicaran la situación.

Sin embargo, y a pesar de tener tantos obstáculos resueltos, la comunicación no resultó fácil entre las partes ya que el cliente estaba en contacto con todo el grupo de desarrollo, hablaba con varios miembros y opinaba sobre como resolver determinadas cosas. Esto sucedía, entre otras cosas porque el cliente mismo era técnico y muchas de las conversaciones surgían por preguntas de implementación que se habían hecho en Argentina. Ese fue uno de los primeros puntos donde se encontró el problema de comunicación, ya que

el grupo de desarrollo sentía que no podía desatender al cliente y se sentía de alguna forma presionado entre el cliente y el responsable de desarrollo local por diferentes directivas e indicaciones.

No solo problemas técnicos surgieron durante la comunicación entre el cliente y el grupo de desarrollo sino problemas de comprensión del dominio en el que se estaba trabajando. Ante la falta de una especificación clara y actualizada, surgían dudas del sistema que eran consultadas directamente al cliente. Así entraron en juego problemas muy profundos como son la diferencia de interpretación de cada una de las personas sobre alguna conversación mantenida o la falta de aviso a todas las partes interesadas en el contenido de las conversaciones. Como consecuencia de ésto, avanzado el proyecto se detectaron inconsistencias sobre la funcionalidad de las partes en el momento de integración y por ende surgieron las diferencias entre los grupos de desarrollo sobre interpretaciones de directivas del cliente. No solo se vió afectado el cronograma de entrega por esta razón, sino que fue inevitable las negociaciones internas y con el cliente para la modificación de estos defectos en el sistema. En ese momento se comenzó a indagar historiales de conversaciones con el cliente y mails enviados y recibidos para aclarar que era lo que el cliente había dicho y estos extractos encontrados en el mail y el chat formaron parte de la documentación del proyecto.

Con la experiencia de un trabajo como el descrito, se comenzó a indagar cómo eran manejadas las comunicaciones en proyectos similares y en base a la experiencia de otros líderes de proyectos se detectó un problema recurrente dentro de la organización. Por ese motivo, se planteó la necesidad de hacer un análisis del problema para abordarlo desde el comienzo de un nuevo proyecto.

El proyecto sobre el cual se comenzó a analizar y definir las comunicaciones era un proyecto nuevo. Con el cliente se había trabajado previamente en el relevamiento y diseño de un módulo de un sistema más general y el proyecto consistía en la implementación de ese módulo. El grupo de desarrollo estaba situado en La Plata y el cliente, también desarrollador de software, estaba situado en Buenos Aires.

La forma de comunicación establecida era vía correo electrónico para las comunicaciones que podían resolverse por ese medio, y vía telefónica si se necesitaba una comunicación para resolver inmediatamente algún asunto relacionado al proyecto. El cliente tenía serias restricciones de seguridad dentro de la empresa, por lo que estaban restringidas las formas de comunicación por otros medios que no fuera el correo electrónico. El Chat y otras formas de comunicación a través de internet estaban restringidas.

Hay una característica fundamental en este proyecto y es la introducción

de un framework de desarrollo muy complejo que el cliente exigía ser usado ya que era de su propiedad. Previo al comienzo del desarrollo se hizo una capacitación sobre el funcionamiento, la arquitectura y los componentes del framework, pero como el mismo era modificado y adaptado constantemente se requería de una actualización contínua sobre estos cambios.

Debido a que el relevamiento y diseño del sistema se hizo muchos meses antes del con el comienzo del desarrollo, partes del diseño que se tenía era ya obsoleto para los componentes del framework. No solo se habían diseñado componentes que no eran adecuados, sino que la modificación o adaptación de esos componentes implicaban un rediseño de las interfaces y del dominio, el cual era muy complejo, y había pocos referentes en el cliente a quien se les podía consultar.

Teniendo en cuenta los problemas del proyecto con la empresa española, el proyecto se organizó de la siguiente forma:

- En el desarrollo del sistema que el cliente había contratado, se establecieron flujos de comunicación. El líder de proyecto era quien hablaba con un contacto en el cliente y los desarrolladores no tenían acceso al cliente de otra forma que no fuera a través del líder.
- Dentro del grupo de desarrollo se definieron los roles correspondientes.
- Ante el surgimiento de un problema grave, si el líder autorizaba, se hacía una consulta telefónica.

De esta forma se organizaron tres formas de comunicación relativamente estructuradas:

- Comunicaciones para la duda de dominios.
- Comunicaciones para dudas de implementación del framework.
- Comunicaciones para el uso del framework.

El proyecto comenzó asumiendo algunos errores cometidos anteriormente e intentando reparar algunos de ellos definiendo los roles y la forma de comunicación a realizar.

Avanzando el proyecto aquello que parecía organizado, terminó necesitando estar en constante ajuste para que funcione. La sobrecarga de trabajo de los desarrolladores que además ahora debían llevar el control de las conversaciones pendientes hizo que no se detectaran las demoras de los tiempos

de respuestas a dudas críticas así como tampoco las dudas que no estaban resueltas. El tiempo empezó a presionar para la entrega del proyecto y varios días pasaron hasta que se empezaron a buscar los motivos de las demoras. Para justificar la falta de capacitación y de respuesta en tiempo, se empezaron a buscar los mails enviados y recibidos.

2.2 Una metáfora motivadora

Existe un recurso que es muy usado en la informática que intenta, a partir de una palabra, sugerir una comparación con algo popular para facilitar la comprensión de un tema particular. Este recurso se llama metáfora y Alistair Cockburn [6] usa este recurso para definir la esencia de un proyecto de software. Esto lo hace en lo que él llama "El manifiesto del juego cooperativo para el desarrollo de software" y dice:

> "El desarrollo de software es un juego cooperativo, en el cual las personas emplean registros como apoyo para comunicar, rememorar e inspirar tanto a sí mismas como a otros participantes con el propósito de llevar a cabo el siguiente movimiento en el juego. El punto final del juego es un sistema de software en operación; el residuo del juego es un conjunto de registros que ayudarán a los jugadores a participar en el próximo juego. El juego siguiente es la modificación o el reemplazo del sistema, o quizá la creación de un sistema vecino."

Define al desarrollo de sofware como como un juego que pertenece a un tipo particular entre todos los juegos debido a que lo encuadra en un juego cooperativo y no competitivo. En este juego no hay competencia entre los jugadores ya que no hay nadie que lo gane o lo pierda, sino que hay cooperación para ayudarse entre todos y todos juntos juegan a la par.

Pero el juego que se juega con un desarrollo de software tiene dos objetivos. El primero es entregar el software y el segundo es sentar las bases y experiencias para lograr mejorar el trabajo del equipo y de sus miembros a través de marcas a eventos a resaltar dentro del proyecto. El sentar las bases y experiencias se aplica durante el proyecto y debería servir también para los siguientes proyectos. Definir lo que en el manifiesto plantea como registros o marcas sirve para:

- Evocar con recordatorios lo que uno ya sabe de alguna discusión o decisión previa. Estos recordatorios deben ser lo suficientemente completos

para recordar los motivos que llevaron a tomar las decisiones en ese momento.

- Hacer marcas para informar al grupo o las partes del grupo sobre diversos sucesos dentro del equipo.
- Por último, sirve como un generador de nuevas ideas.

Lo que plantea Alistair Cockburn [6] es ver al desarrollo de software como un juego y como una oportunidad donde registrar los sucesos dentro del mismo para asentar el aprendizaje del juego. Este aprendizaje debería ser considerado inmediatamente en el juego así como también en un futuro en otro nuevo juego.

La escencia del juego planteado es el trabajo en equipo y para que un equipo funcione debe haber comunicación y cooperación. Esta tesis se centra en los aspectos de comunicación del juego focalizando en que dentro de las comunicaciones se encuentran los hitos y sucesos desde donde el grupo tiene que aprender para lograr los objetivos comunes.

Esta metáfora será usada en la siguiente sección para motivar la búsqueda de los requerimientos necesarios para lograr una productiva comunicación en el equipo.

2.3 Los objetivos a cumplir por el modelo que se propone

Hay muchos factores que inciden en el éxito de un proyecto y son tan amplios que pueden variar desde la capacidad técnica del grupo o la integración del mismo hasta el trato y la forma de negociación con el cliente. En todos estos factores la comunicación está presente y el éxito en estas comunicaciones garantiza el éxito del proyecto.

Para lograr ganar el juego es esencial organizar el equipo y para esto se usan los roles. Cada tarea a realizar dentro del desarrollo de software implica la definición de un grupo que lo lleve a cabo y pueden hasta generarse subgrupos dentro de ellos para organizar el equipo. Así, cada persona tiene definida su contribución al juego y tiene una identificación en el mismo. Para esto se necesita que, por un lado, se administren a cada una de estas personas como una entidad única dentro de las conversaciones, y por otro se necesita que se identifiquen bajo el nombre común de los grupos o roles a los que

representa. Ambas necesidades son requerimientos que serán denominados R01 y R02 respectivamente.

Con la idea de que se necesita cooperar y trabajar en equipo y focalizando en el objetivo de cumplir el objetivo final de concluir exitosamente el juego, las comunicaciones dentro del equipo serán de dominio público (a este punto se lo referenciará como R03 por ser el primer requerimiento funcional del sistema). Además, las comunicaciones se resolverán a partir del concepto de conversación, conviniendo que una conversación es una serie de comunicaciones relacionadas que mantienen dos o mas personas con el objetivo de negociar, entender o convenir un tema particular (R04).

Una vez que se definieron los roles y las conversaciones, es necesario analizar como debería ser el flujo de estas últimas. Analizar este flujo contribuye a organizar y planificar como sería la mejor forma de funcionamiento en equipo para que la información llegue a todos los miembros que tienen alguna vinculación con la misma. Es decir, que llegue en tiempo y forma a quienes deben manejar esa información. Definir estos flujos, permite que se establezcan circuitos de comunicación para poder controlar, supervisar e informar sobre los temas en cuestión (R05).

Cada conversación representa la forma de trabajo del equipo. Dentro de la misma se pueden analizar decisiones o definiciones. Tener disponible en todo momento y lugar cada conversación hace que no se pierda la historia de la interacción y del trabajo del grupo, por lo que será necesario contar con la posibilidad de recuperar siempre estas conversaciones (R06). Como algunas conversaciones representan toma de decisiones o resolución de algún conflicto, es necesario poder hacer un seguimiento sobre el estado de las mismas. Para esto es imprescindible no solo poder disponer de estados que indiquen si la conversación está en curso o cerrada, sino también poder contar con una fecha de espera de respuesta por parte de quien genera la conversación (R05). Esto último busca como objetivo poder contar con un mecanismo que permita agilizar las conversaciones además de poder lograr hacer un seguimiento de temas pendientes o conversaciones por resolver (R08).

Por otro lado, en un proyecto de software, muchas veces se generan debates internos para resolver de distintas formas un diseño o un aspecto técnico. Para lograr esto, es necesario que se definan métodos de conversación grupal que permitan estas discusiones(R09).

Debido a que las conversaciones representan la comunicación del equipo, en ellas se pueden ver las discusiones de diseño, la toma de decisiones del equipo y la forma de funcionamiento e interacción del mismo entre otras cosas. Es entonces necesario que las mismas estén accesibles públicamente

y puedan ser listadas (R10), navegadas (R11) y exportadas (R12) para su publicación y distribución. Además, deberán poder ser marcadas y / o catalogadas para lograr los objetivos que se presentan en el manifiesto de Alistair Cockburn [6] (R13).

A lo largo del proyecto pueden surgir conversaciones que traten de un tema particular o que repitan un tema o una decisión tomada. Para lograr un registro de la vinculación entre ellas, es necesario contar con un mecanismo que las vincule (R14).

Cuando el juego se de por concluido, será necesario que el equipo completo analice el juego para poder aprender y terminar de conocer cómo funcionaron como equipo. Para esto, se deberá proveer un mecanismo que permita hacer notas sobre las conversaciones y así registrar comentarios post mórtem del proyecto (R15).

2.4 Conclusiones

En este capítulo se presenta el escenario concreto sobre el que se trabaja y se presentan todos los requerimientos que la solución a desarrollar debe cumplir. A modo de resúmen y de referencia rápida a continuación se listan todos estos requerimientos.

- R01: Administración de usuarios: Se debe permitir el alta y modificación de usuarios que represente cada uno de los miembros del equipo.
- R02: Mecanismos de abstracción de usuarios: Se debe proveer un mecanismo de abstracción para representar los roles dentro del equipo.
- R03: Conversaciones públicas: Se debe mantener un registro de todo lo conversado y debe ser accesible publicamente.
- R04: Conversación como unidad de trabajo: Se debe permitir la vinculación de todos los mensajes en una conversación y la consulta de todas las conversaciones generadas.
- R05: Se debe permitir definir flujos para las conversaciones: restringiendo o permitiendo comunicaciones entre grupos de trabajo o roles distintos.
- R06: Se debe permitir que en cualquier momento cualquier mensaje pueda ser recuperado.

- R07: Se debe permitir determinar tiempos para las conversaciones: cada conversación puede contar con un tiempo límite de espera de cierre de la misma.
- R08: Se debe permitir el seguimiento de las conversaciones abiertas.
- R09: Se debe permitir definir conversaciones grupales según los mecanismos de abstracción definidos.
- R10: Se debe permitir el listado de las conversaciones que sufrieron demoras.
- R11: Se debe permitir la navegación y búsqueda de conversaciones.
- R12: Se debe poder exportar las conversaciones como un documento.
- R13: Se debe poder marcar y clasificar las conversaciones.
- R14: Se debe permitir vincular conversaciones asociadas.
- R15: Se debe permitir hacer anotaciones sobre las conversaciones en forma de análisis post mórtem del proyecto

Capítulo 3

Estado del arte

En este capítulo se presentan las herramientas mas representativas que existen actualmente y que dan soporte a las conversaciones basadas en computadoras. Todas las herramientas y aplicaciones se presentan para analizarlas desde un punto de vista de utilidad para cubrir los requerimientos antes mencionados.

3.1 Sistemas de escritura colaborativa

La idea de la escritura colaborativa es proveer un espacio común donde un grupo de personas pueda escribir sus ideas. Se puede pensar este espacio como un pizarrón donde cada uno puede escribir o como un documento. La escritura sobre este espacio puede ser sincrónica o asincrónica. Así, en un pizarrón compartido y con alguna herramienta adicional para comunicación se puede debatir sincrónicamente mientras que se va dibujando sobre el pizarrón. La alternativa asincrónica, puede pensarse como un espacio en donde escribir y dejar disponible para que en otro momento otra persona agregue información o modifique la misma.

Dentro de los sistemas de escritura colaborativa, una herramienta muy aceptada es la Wiki [36]. Esta herramienta ideada por Ward Cunningham está implementada en muchos lenguajes y existen actualmente implementaciones de código abierto. El objetivo de la misma es en forma fácil, sencilla y rápida , modificar y publicar sitios web. Cualquier otra persona que tenga acceso a la página resultante, puede editarla, comentarla y extenderla. Es una forma muy sencilla de trabajar asincrónicamente colaborando en un grupo de personas y no requiere que se conozca la sintaxis de html ya que tiene una sintaxis propia muy simple que la herramienta misma se encarga de traducir a html.

Otras de las herramientas a destacar, son los procesadores de texto con control de cambios, revisión y anotaciones. Estas herramientas ayudan a crear documentos. Mediante controles de cambios y la posibilidad de agregar notas, sumados al sencillo manejo de versionamiento que algunos programas tienen incoroporados hacen que se transforme en una herramienta para el trabajo colaborativo. Por lo general también son herramientas asincrónicas ya que permiten que solo una persona a la vez edite el documento. Ejemplos de estas herramientas son MS Word [26] y el editor de texto de Open Office [28].

En las herramientas asincrónicas no se da soporte para que el grupo de trabajo interactúe o converse mediante el documento. Así, las conversaciones relacionadas al documento son implícitas en el contenido del documento mismo (por ejemplo cuando en el documento se hacen anotaciones entre los miembros del equipo), o explícitas, con otras herramientas que soportan la comunicación.

En los casos en que se busca escribir un documento colaborativamente y se necesita un soporte explícito, se puede usar un documento sobre el cual una persona escribe mientras el resto del grupo puede ver lo que va escribiendo y una herramienta de chat o voz que soporte la conversación de las partes colaborantes.

Si bien las herramientas presentadas son de utilidad para muchas situaciones, cualquiera que quisiera buscar dentro del documento debería leerlo para ver su contenido a menos que se genere un índice o que alguien organice el documento. En los casos de las Wiki [36] ocurre algo similar que puede resolverse con un soporte de búsqueda en las páginas. Igualmente, la necesidad de tener acceso a internet es restrictivo al uso de la misma para el escenario planteado ya que en el escenario la navegación http es restringida.

3.2 Voz/Video

Las herramientas de comunicación vía voz o video, proveen una comunicación entre dos o mas personas. Permiten mediante computadoras conectadas (en una red local o a través de internet) la transmisión de voz, de video o ambas simultáneamente.

Las comunicaciones de este estilo, son las que más se asemejan al trato personal, por lo que su estructuración o forma de interacción puede ser vista de la misma forma. Es decir que por lo general se caracterizan por:

- Una persona habla a la vez.

- Cuando alguien habla, no es interrrupido.
- Pequeños silencios son los momentos donde otra persona puede pedir hablar.

Si bien esta es la forma más común de comunicación para voz o video, pueden plantearse mecanismos alternativos como por ejemplo una herramienta de voz con turnos, donde una persona puede tener permiso para hablar y al terminar le pasa el permiso a quien le siga en turno entre tantas otras formas.

Estos protocolos o estructuración de las conversaciones pueden ser provistas por la propia herramienta o coordinarse entre las partes participantes para lograrlas. Un análisis sobre la mejor forma de trabajo en el grupo sería necesario para definirlo.

A continuación se listan las siguientes dificultades que pueden asociarse a estas herramientas:

- Si la forma de transmisión de conocimiento es mediante la grabación, la búsqueda de información es secuencial. Para encontrar parte de la conversación se debe escuchar linealmente toda la conversación.
- Si la forma de transmisión de conocimiento es mediante una minuta de reunión, implica tiempo para escribir esa minuta de reunión, compromiso de las partes en verificar la misma y tiempo adicional para dejar la información disponible en un lugar común a todo el proyecto.En este caso, no se podría resolver sencillamente la clasificación de los temas tratados en la miniutas ni poder destacar temas importantes tratados en las mismas.
- Es una forma de comunicación sincrónica, por lo que se necesita que las partes concuerden para llevar a cabo la reunión. Esto es complejo de lograr en un proyecto distribuído con distinta carga horaria, pero en el escenario planteado esto no sería un obstáculo.
- Es una de las herramientas mas fuertemente restringida y resistidas a habilitar dentro de muchas empresas, debido a políticas de seguridad y para evitar la distracción de sus empleados.

En el escenario planteado existe una fuerte restricción de acceso a internet y este tipo de herramientas no son permitidas, por lo que no son adecuadas para el problema planteado.

3.3 Chat

Las herramientas de chat, permiten escribir mensajes que son enviados a una persona o a un grupo de personas. Existen herramientas que permiten el envío de mensajes sólo cuando las partes están presentes como el MSN Messenger [25], mientras que otras posibilitan el envío de mensajes aunque el destinatario esté ausente como por ejemplo el ICQ [17].

Estas herramientas tienen algunas diferencias entre ellas, como por ejemplo:

- Historia de las conversaciones: Algunas herramientas permiten guardar las conversaciones en una forma de registro con fechas / horas de envío.
- Sesiones de voz: Permiten comunicaciones por voz además de la conversación escrita.
- Contextualización: Actualmente todos los chats proveen formas de dar contexto a las conversaciones sobre estados de ánimo dentro de la charlas mediante representación gráfica o de símbolos. También proveen información del estado de la persona en la conversación: si la persona con quien se está hablando se encuentra escribiendo en la máquina o si por un momento se ausentó de la misma. A toda esta información, entre otra, se la conoce con el término en inglés awareness, término que se usará de ahora en adelante.

Las herramientas de chat mas populares, como el Yahoo! Messenger [39] el MSN Messenger [25] o el ICQ [17], no tienen forma de estructuración de conversaciones. En cualquier momento se puede escribir y se pueden mantener varias conversaciones simultáneamente. Al igual que la voz o el video, se puede acordar la estructura de conversación a mantener entre las partes.

Aunque actualmente son muy usadas las herramientas de chat, existe aún una gran resistencia a las mismas por resultar de alguna forma invasivas en el trabajo cotidiano de algunas personas. Al margen de este problema que puede surgir en algunas personas, también se encuentran las siguientes dificultades con esta herramienta:

- Si se conversa desde distintas máquinas, por lo general, el log no está centralizado, por lo que buscar una conversación en todas sus partes puede implicar tener que buscar en todos los logs de todas las máquinas donde se estuvo chateando.

- Si los logs del chat forman parte de la documentación del proyecto como se vio en los capítulos anteriores que sucedía, entonces los mensajes pueden ser difíciles de entender sin el contexto apropiado cuando los mismos son releídos por otras personas
- En el chat muchos hilos de conversación pueden intercalarse mientras se mantiene una conversación, lo que dificulta mas aún ver a los logs como una forma de transmisión de conocimiento.
- Al igual que la voz y video, son herramientas restringidas dentro de muchas empresas no solo por seguridad, sino para evitar la distracción de los empleados.

A pesar de que algunos de estos problemas pueden ser resueltos con alguna modificación en el funcionamiento o con un acuerdo previo entre las partes de cómo conversar, es inevitable la resistencia que tienen estas herramientas dentro de las empresas. En particular, para el escenario planteado se hace imposible su uso.

3.4 Correo electrónico

El correo electrónico o e-mail es la herramienta colaborativa más aceptada. La simpleza y la semejanza con el tradicional envío de correo ha transformado a esta forma de comunicación en una de las más populares.

La forma de comunicación con el correo electrónico es asincrónica, por lo que permite que las partes que conversan puedan tomarse el tiempo necesario para responder a un correo. Esto ayuda a que las personas que tienen ciertas resistencias a lo invasivo que puede ser el chat, encuentren en el correo electrónico una forma mas adecuada a sus necesidades de comunicación.

La estructuración de las conversaciones en los correos electrónicos es sencilla. Las conversaciones se vinculan entre sí a partir de cierto contenido que los clientes de correo embeben en cada reenvío de un mensaje o en la respuesta al mismo. Por ejemplo, muchos usan el símbolo "Re:" para indicar que es una respuesta a un correo. Además, según el cliente de correo electrónico que se tenga, se pueden encontrar los mails relacionados en una forma muy básica a partir de su asunto. Hasta no hace mucho, esta posibilidad se daba solo en los clientes de mail stand alone pero actualmente GMail [10], el servicio de correo electrónico de Google [12], provee una forma de leer el correo electrónico desde la web y el propio site agrupa los correos en conversaciones automáticamente.

El correo electrónico es la herramienta menos restringida en las empresas. Sin embargo, muchas de ellas permiten sólo el acceso al servidor de correo de la misma empresa, impidiendo el acceso a otros posibles correos personales que tengan sus empleados.

Existen casos donde las empresas mismas por cuestiones de seguridad restringen el acceso a algunos correos electrónicos con datos adjuntos. Esto se debe a que si bien el correo es seguro, sus archivos adjuntos pueden no serlo y hay políticas donde no aceptan archivos adjuntos y por ende los eliminan o el propio sistema de seguridad verifica el contenido del mismo y en caso de ser dudoso lo elimina. Esto pasa cuando algunos sistemas antivirus no pueden determinar la ausencia de virus y podan el correo dejando solo la parte segura.

El correo elctrónico por ser asincrónico, permite que los tiempos de respuestas sean manejados por el destinatario del correo. Es decir, que los tiempos de respuestas los pueden dilatar las partes según su conveniencia o necesidades.

Sobre la estructuración de comunicaciones caben destacar los siguientes problemas que surgen con el correo electrónico:

- El envío de correo puede hacerse desde distintas máquinas y no hay un servidor central con todos los correos, por lo que no es posible mantener todas las conversaciones completas.
- Es frecuente, dependiendo sobre todo de la organización del receptor, olvidarse de responder mails o que los mismos se traspapelen.
- Es muy fácil de alterar la estructuración de las conversaciones solo por el agregado del Re: en el asunto.

3.5 Listas de correo y grupos de noticias

En esencia los correos electrónicos, las listas de correos y grupos de noticias son similares. Una persona escribe un correo electrónico pero en lugar de indicar una dirección personal indica una dirección que identifica una lista y lo reciben todas aquellas personas que estén suscriptas a la misma. Lo que diferencia las listas de los foros es que la suscripción a una lista implica que cuando un mensaje es enviado, todos los miembros reciben el mensaje, mientras que con un grupo de noticia los mensajes son enviados y los miembros buscan los mensajes cuando lo solicitan, es decir, bajo demanda.

En estas herramientas, todos los mensajes son públicos y son accedidos por todos los miembros, por lo que pueden haber mediadores para los contenidos de los mensajes. Además, por ser públicos, deberían ser específicos para lo que el grupo se suscribe y para que la audiencia reciba lo que espera.

Un grupo de noticias no es un ejemplo de conversación porque sólo se reciben noticias. En cambio una lista tiene la misma estructuración de conversaciones de correo electrónico con la diferencia de que lo escrito es público.

Pensando en los problemas planteados que hay que resolver los grupos de noticia podrían ser útil para informar sobre distintas novedades descubiertas sobre algún aspecto que le interese al grupo de trabajo, pero implica el compromiso de los integrantes de ese grupo de leer los contenidos. Además, se necesita una herramienta adicional para consultar esas novedades y así evitar que sean a través de la búsqueda de mails el compartir conocimiento con integrantes nuevos. Algo similar pasa con las listas, pero ambas comparten los siguientes problemas:

- Todos quienes están susriptos a la lista o grupo reciben los mails
- Se tiene que coordinar en la información que circulará en el grupo o la lista para que los contenidos le lleguen solo a los interesados.
- Si existen distintas especializaciones, implica la creación de distintos grupos o listas para que lleguen a la audiencia adecuada.
- Si hay muchos mensajes, terminan siendo tediosos para seguirlos y pueden que dejen de leerse

3.6 Workflows de comunicación

Un Workflow es una herramienta que permite describir en un sistema procesos predefinidos de una organización para lograr un fin común dentro de ella. Así, mediante el concepto de nodos y links se pueden representar estados o subestados en el que algún proceso puede estar (representado por los nodos) y a cuales otros estados o subestados puede seguir (los links).

Hay herramientas que tienen este flujo predefinido para actividades muy generales como pueden ser aquellas analizadas por Terry Winograd [38] para las teorías de discursos. Este workflow no puede ser cambiado y es estricto en el sentido de que no se puede configurar otro workflow. Otras herramientas pueden ser definidas para un objetivo particular pero con cierta flexibilidad

a la hora de definir el workflow. Como ejemplo de esta última, se puede mencionar a Bugzilla [3] que es una herramienta particular para el seguimiento de bugs dentro de un proyecto de software. Esta herramienta permite, dentro del contexto de seguimiento de bugs, definir el circuito de corrección según las preferencias del grupo de desarrollo. Otra herramienta de workflow para igual funcionalidad es JIRA [8].

Hay ejemplos donde pueden aplicarse workflows y todos ellos pueden verse como una forma de comunicación dentro de un equipo de trabajo a partir de la transición de estados dentro del workflow.

Actualmente hay aplicaciones de código abierto que permiten diseñar estos workflows para cualquier aplicación general.

3.6.1 El Coordinador

Terry Winograd [38] habla de una perspectiva de lenguaje-acción y analiza una herramienta llamada "The Coordinator" que se basa en dicha perspectiva. En una forma simplificada, Winograd presenta un ejemplo simple de lo que es una conversación para la acción, presentándola mediante el flujo de estados de la Figura 3.1.

En la Figura 3.1 se muestra como cualquier conversación para la acción puede generalizarse en el workflow planteado. Si una persona A, hace un pedido a B, esta persona puede prometerle tener respuesta para una fecha dada. En algún momento esta persona B, reporta completado el pedido y A puede aceptar el resultado. Las posibles alternativas a esta conversación son los demás posibles circuitos que pueden formarse en el recorrido del workflow.

"The coordinator" [7] es una herramienta que, entre otros esquemas de conversaciones, implementa esta estructuración de conversación para la acción. Desarrollada en los años 80', permite utilizar un servicio de mensajería al estilo del correo electrónico. Si bien no es puramente un workflow, es mas sencillo mostrarlo como si lo fuera para ver gráficamente el circuito de comunicación.

Esta herramienta permite ver qué conversaciones una persona puede tener pendiente (es decir las conversaciones que no están en un estado de finalizado como las marcadas en el dibujo en los nodos 5, 7, 8 y 9), las conversaciones que hoy tienen vencimiento de reporte (en las cuales la promesa de fecha de respuesta es el día de la fecha) o las conversaciones que aún no han sido respondidas, por enumerar las funcionalidades mas significativas.

A pesar de ser una buena solución para estructurar conversaciones, en un análisis hecho por Thomas Schäl [29] se presentan los resultados de la

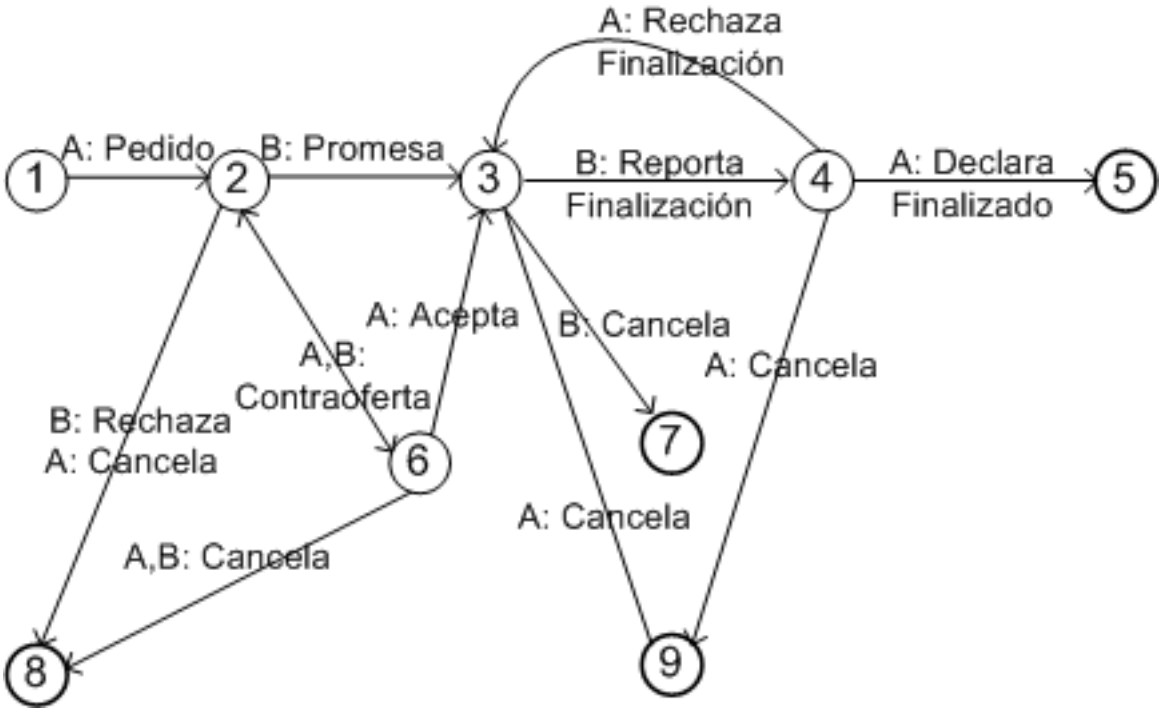

Figura 3.1: Circuito de conversaciones para la acción.

observación y relevamiento de un grupo de trabajo en Italia que usaba el sistema. Este es un resumen de ciertas conclusiones del trabajo:

- Se reconoce la potencia de la herramienta, pero se usa como simple medio de comunicación convencional. Es decir, se detectó que cierta gente es conciente de los beneficios que pueden obtenerse con ella, pero pocos usan realmente a la misma en su conjunto.
- Se reconoce la necesidad de plantear roles dentro del trabajo para que a partir de eso pueda usarse la herramienta. Una conversación para la acción siempre concluye que una de las partes tiene que hacer trabajo para la otra parte. Si los roles no están bien definidos, esto puede conducir a ciertos problemas entre las partes que tratan.
- Se reconoce la necesidad de tener una definición de procesos de trabajo para su uso.
- Se reconoce la necesidad de una fuerte capacitación para el uso de la herramienta.

Capítulo 4

Resultados tempranos

En este capítulo se describe una primera solución brindada al grupo de trabajo para solucionar los problemas de comunicación para el escenario planteado. En particular, se cuenta cómo se resolvió la organización de las comunicaciones y cuál fue la experiencia de la misma.

Lo que en este capítulo se muestra, forma parte de la maduración y la evolución de la idea de esta tesis. Además, se muestran muchas de las ideas que incentivaron a la búsqueda y profundización de una solución a un problema recurrente dentro de los proyectos de software.

4.1 Introducción

La introducción de un software o una nueva metodología de trabajo para grupos de personas es un área bastamente investigada dentro de las áreas de Groupware y CSCW. Es imprescindible que una vez introducida la forma de trabajo se analice constantemente la nueva forma que este trabajo adopta ya que es contínua la necesidad de adaptación y transformación entre la metodología y el grupo de trabajo.

La necesidad de evolucionar y adaptarse entre las partes que interactúan amoldándose y transformándose se conoce con el término de coadaptación [1]. Esta coadaptación es lo que se comentará brevemente en este capítulo mostrando como una solución inicial dentro del grupo de trabajo provocó la adaptación constante entre las partes del grupo de trabajo y la metodología de trabajo adoptada.

[1] El término coadaptación es usado en varias ciencias. En particular, en biología el término coadaptación es aplicado para explicar como se adaptan mutuamente especies, genes y partes de organismos como parte de una evolución conjunta entre las partes que se adaptan.

El grupo de desarrollo involucrado en el escenario planteado, ya siendo conciente de las implicancias de la falta de organización en las comunicaciones, intentó abordar el proyecto proveyendo una solución sencilla al problema. Dentro de las alternativas de solución que surgían bajo las limitaciones que el escenario tiene, se esbozó una solución inicial.

Primeramente en este capítulo se introducirá la estructuración de conversaciones que otro grupo de desarrollo de la misma organización tenía para las conversaciones relacionadas al manejo de versionado de código. Esta primera parte tiene dos objetivos: mostrar que las conversaciones pueden estructurarse desde muchos aspectos en un proyecto de software y por otro lado, mostrar qué inspiró al primer diseño de solución del escenario planteado.

Seguidamente, se mostrará el diseño inicial de la solución al problema y las consecuencias y adaptaciones que este diseño tuvo a lo largo del proyecto.

4.2 Parte I: Organización para uso de repositorio de código

Dentro de uno de los grupos de desarrollo analizados en la investigación de esta tesis hubo uno en particular donde se encontró en forma explícita la estructuración de conversaciones.

El problema del grupo de desarrollo se centraba en organizar ciertas conversaciones que ocurrían en torno a la administración del código fuente que desarrollaban. En el grupo de desarrollo de aproximadamente diez personas, nueve de ellas eran desarrolladores de software y la décima cumplía el rol de coordinación del resto del grupo. Si bien el grupo de desarrollo era mas extenso, solo estas diez personas son relevantes al ejemplo que se cita.

El grupo de desarrollo trabajaba con versionamiento de código mediante un CVS [8] (llamado CVS por el inglés Concurrent Versions System) que permite tener una versión centralizada del código en un servidor y que cada desarrollador desde su lugar de trabajo cuente con una copia de la misma y trabaje localmente sobre ella. Existe entonces una necesidad de coordinar el momento donde se aplican los cambios de todos los desarrolladores sobre la versión centralizada y para esto se necesita estructurar las conversaciones que puedan surgir.

Cuando el proyecto comenzó, se realizó una especificación del plan de administración de configuración según lo establecido en CMM-Nivel 2 [5] de SCM (del inglés Software Configuration Management que significa adminis-

tración de configuración de software). En este plan, se estableció la forma de versionado y el circuito de notificaciones para organizarlo.

Definieron así distintos templates de mail, cada uno con una finalidad específica que se muestran en las siguientes figuras.

En la Figura 4.1, la primera de las tres conversaciones, se muestran los pasos a seguir para informar de la creación o modificación de un documento. Quien inicia la comunicación es un usuario que necesita notificar que ha creado o modificado un documento y lo ha incorporado al CVS [8] (paso a.). El template del mail que se debe enviar para esta notificación se muestra en la Figura 4.2. Este mail le es enviado al encargado de SCM quien junto a gente de QA (Quality Assurance, equipo de control de calidad) verifica los aspectos relacionados a ambas tareas. Una vez revisado el documento, se le envía un mail de respuesta con el template que se muestra en la Figura 4.3 (paso b.). El mismo puede ser un mensaje de aceptación o de rechazo. En este último se puede especificar las razones y eventualmente pedir modificaciones para la posible aceptación

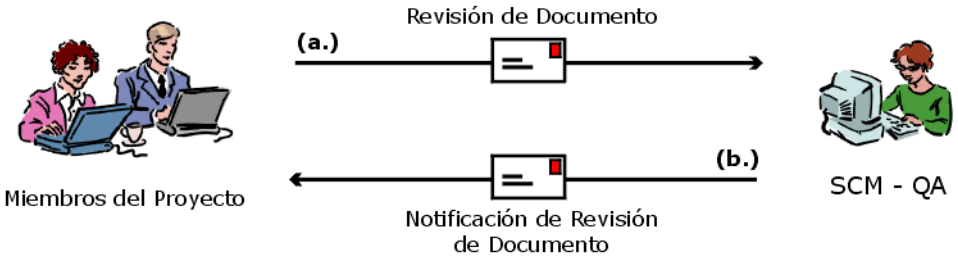

Figura 4.1: Conversación por documentos.

En la Figura 4.4 se muestra la segunda conversación que son los pasos a seguir para informar de la creación o modificación de componentes de código.

Inicia la conversación el desarrollador que publicó el nuevo código en el CVS [8] (paso a.). Este mail se muestra en la Figura 4.5 y va dirigido al administrador de SCM quien verifica los aspectos relacionados con su área y realiza una revisión con QA. Según los resultados a los que arriben, pueden mandar un mail con el template de la Figura 4.6 o de la Figura 4.7. El primer template se envía si los cambios fueron aceptados. El segundo se envía si los cambios se rechazan y puede incluir las razones por las cuales se rechaza.

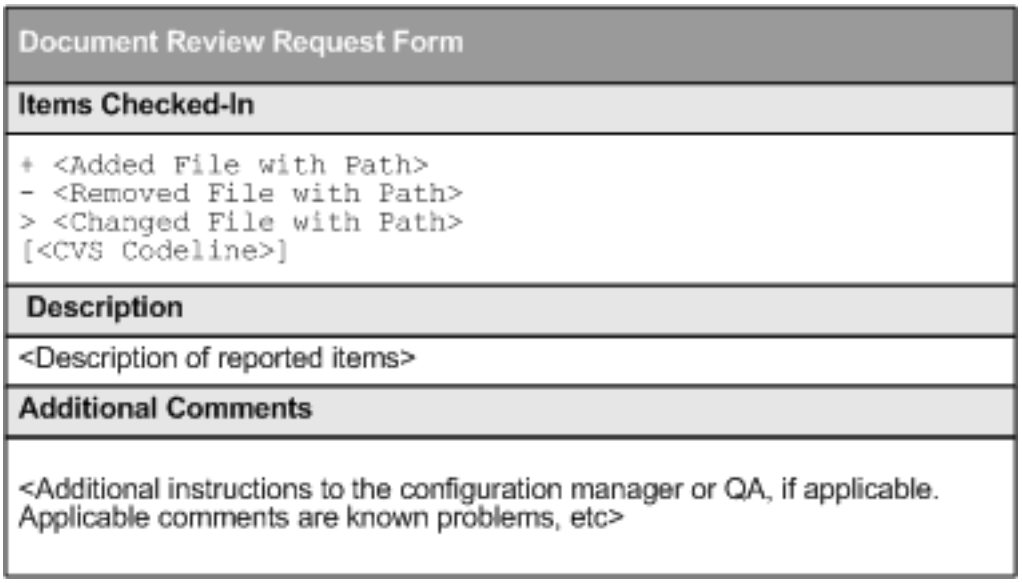

Figura 4.2: Formulario de revisión de documento

En algunos de los tempates de mails hay una referencia a un link de JIRA [21] que le da contexto a la tarea que tiene asignado el desarrollador. JIRA es una herramienta adicional que se usa para la asignación de tareas a cada desarrollador. Mediante este link se puede ver a que hacen referencia los cambios introducidos.

Por último, la tercera conversación y la mas simple es la que se muestra en la Figura 4.8. El único fin de esta conversación es informar al administrador de SCM que un nuevo proyecto se ha agregado al CVS [8]. El template del mail se muestra en la Figura 4.9.

4.2.1 Los problemas detectados

Esta forma de administrar los cambios, resulta ser muy organizada y por sobre todas las cosas está bien definida. En un primer intento por resolver estas conversaciones, se puede decir, que este esquema satisface los objetivos buscados:

- Los canales de comunicación están bien definidos: aquellos que estén vinculados al proyecto deben leer tanto el SCM como el PP (de Project Plan o plan de proyecto) entre otros y por ende les debe quedar claro como trabajar con el CVS [8].

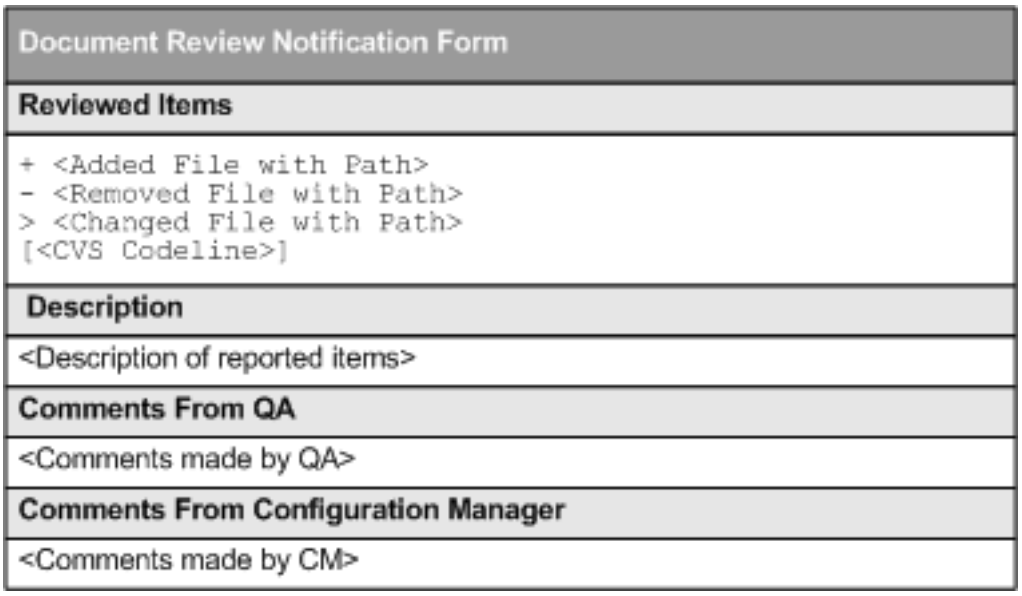

Figura 4.3: Formulario de respuesta a una revisión de documento

- La herramienta de trabajo está establecida: se define el correo electrónico como herramienta de trabajo y se proveen los templates correspondientes para enviar los mensajes.
- Un orden se establece y el circuito se logra controlar: todos los documentos que se suben al CVS [8] están verificados por el administrador de SCM y por QA.
- Los circuitos fueron bien aceptados por los desarrolladores: esto se debe a que estaba unificado el sistema de mail que se usaba en la empresa y los mails pudieron armarse como templates de ese mismo sistema. Estos dos puntos sumados a que el mail era una de las herramientas mas usadas por el grupo de trabajo, llevaron a que no haya gran impacto en la implementación de esta metodología de trabajo.

Por otro lado, hay ciertos objetivos que no se lograron:

- Los circuitos de las conversaciones no se usaron en forma completa ya que hubieron templates, como el de rechazo de código, que nunca fue usado.
- Lo que se pensaba que fuera una herramienta de análisis, no resultó como tal. Esto fue debido a que los tiempos hicieron que no se trabaje exhaustivamente con QA. Como consecuencia de ésto no se detectaron

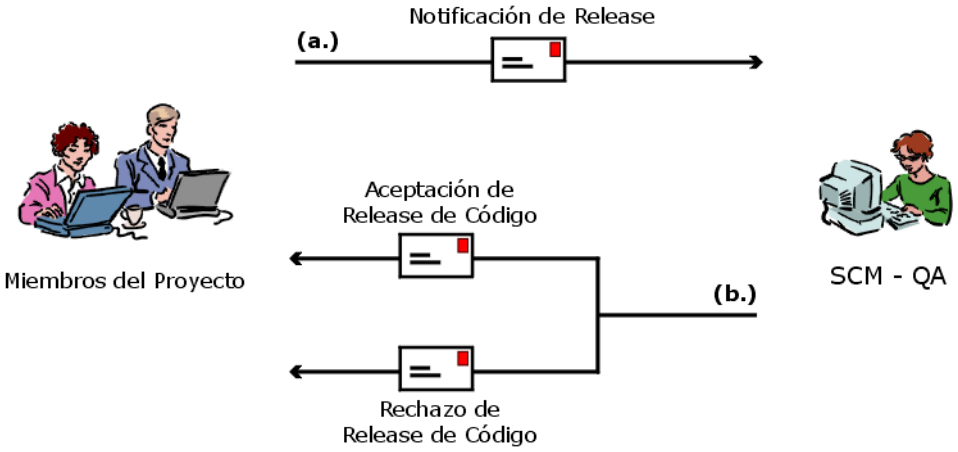

Figura 4.4: Conversación por releases.

fallas dentro del código y por esto ciertos templates no fueron usados. Por lo tanto no se pudo tomar al circuito de mails como una forma de análisis del funcionamiento del grupo de trabajo.

4.3 Parte II: Organización del diálogo con el cliente

Como se comentaba previamente en la Sección 2.1, el grupo de desarrollo estaba ubicado en La Plata mientras que el cliente se encontraba en la Ciudad de Buenos Aires y la comunicación inicialmente se estableció únicamente vía correo electrónico o telefónicamente en caso de urgencias.

Mediante un framework desarrollado por el cliente, se necesitaba implementar un par de módulos para un sistema mayor de telefonía. Este framework fue por primera vez instalado fuera del cliente para desarrollar en La Plata, con la particularidad que se había cambiado radicalmente de plataforma ya que se instalaba por primera vez en MS Windows tras haber trabajado siempre bajo un sistema operativo UNIX.

A lo largo del análisis de este grupo, se presentaron diferentes problemas en distintos aspectos de lo que fue su desarrollo. Así, los problemas que surgían eran sobre dudas del dominio, sobre configuración del framework, sobre funcionamiento del mismo y hasta de diferencias de versiones entre el framework instalado en La Plata y las distintas evoluciones y versiones que se manejaban en Buenos Aires.

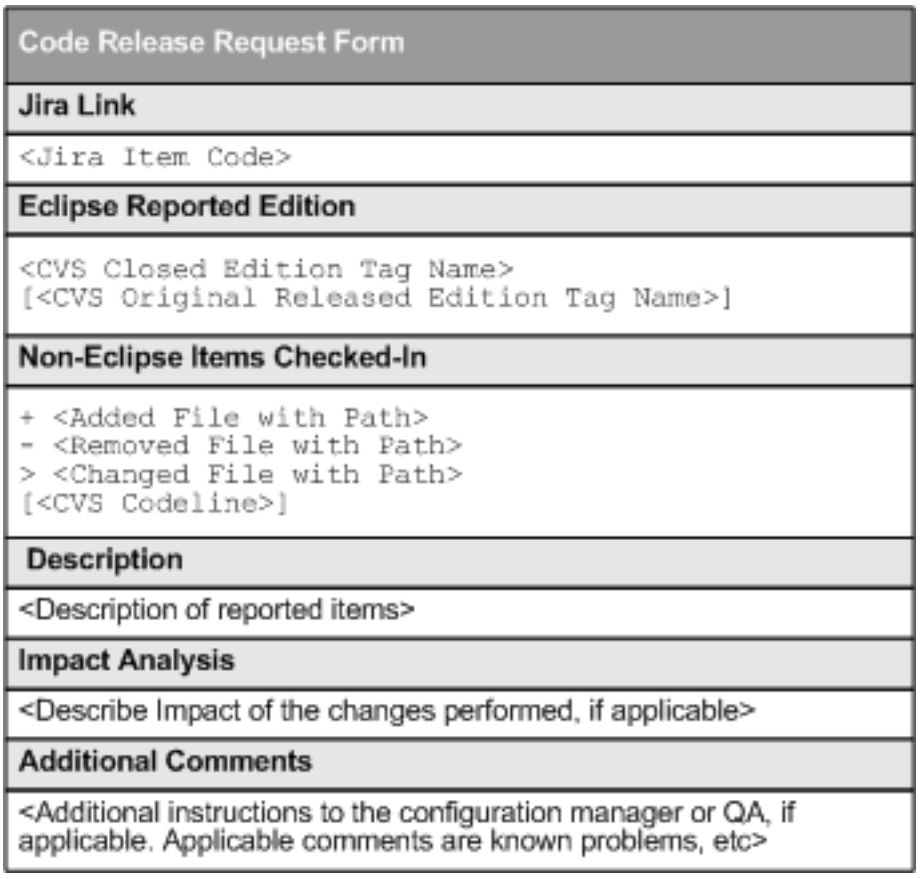

Figura 4.5: Formulario de informe de release de código.

El proyecto estuvo organizado desde el principio y en constante adaptación ante los problemas que surgían. Cuando comenzó, los roles se definieron explícitamente para cada integrante, y a medida que se iba mejorando la forma de trabajo, los roles iban adaptándose para seguir manteniendo la armonía de trabajo.

En las próximas subsecciones se relatan las distintas evoluciones en lo que respecta a la estructuración de conversaciones.

4.3.1 Definición de la estructura de comunicación

El primer rol que fue necesario cubrir fue aquel que se encargara de la comunicación con el cliente. Si bien el grupo de desarrollo había sido capacitado por el cliente para el uso del framework y el mismo había sido instalado por el cliente, muchos problemas se presentaron por lo que se necesitó una formalización de las consultas al mismo. El líder del proyecto, adoptó entonces

Code Release Notification Form
Jira Link
<Jira Link>
Eclipse Reported Edition
<CVS Released Edition Tag Name>
Eclipse Related Teamset
<Hyperlink to the team set (in ViewCVS) into which the notified changes were included>
Non-Eclipse Released Items
+ <Added File with Path> - <Removed File with Path> > <Changed File with Path> [<CVS Codeline>]
Comments From QA
<Comments made by QA>
Comments From Configuration Manager
<Comments made by CM>

Figura 4.6: Formulario de informe de aceptación de release de código.

el rol de comunicador con el cliente.

Forma de comunicación establecida: Si alguno de los desarrolladores tenía algo para preguntar sobre el framework entonces debía enviar un e-mail al líder, este lo verificaba y ampliaba (en caso que se necesitara mayor contexto para el problema encontrado) y se lo enviaba a algún miembro del cliente según el tipo de pregunta (dominio o tecnología entre otras). Cuando desde el cliente, se obtenía una respuesta, el líder se encargaba de reenviar la respuesta a quien correspondiera dentro del grupo de desarrollo.

4.3.2 Primer cambio en las conversaciones

Con la forma de comunicación establecida, surgieron otras necesidades como la de descubrir que consultas estaban pendientes, cuáles estaban demoradas y cuáles habían sido resueltas. Además, comenzaron a surgir dudas reiteradas, por lo que era necesario que las respuestas enviadas desde el cliente estén

Code Rejection Notification Form
Jira Link
<Jira Link>
Eclipse Reported Edition
<CVS Closed Edition Tag Name> [<CVS Original Released Edition Tag Name>]
Non-Eclipse Items Checked-In
+ <Added File with Path> - <Removed File with Path> > <Changed File with Path> [<CVS Codeline>]
Comments From QA
<Comments made by QA>
Comments From Configuration Manager
<Comments made by CM>

Figura 4.7: Formulario de informe de rechazo del release de código.

disponibles en un lugar común.

Una de las fallas detectadas fue que las personas a quienes se contactaba en el cliente, no eran las adecuadas o simplemente no respondían en tiempo los correos electrónicos. Este último caso fue uno de los mas detectados, ya que había cierta resistencia en "perder el tiempo" respondiendo mails cuando sus jefes no estaban al tanto de las consultas que se le hacían.

Forma de comunicación establecida: Si alguno de los desarrolladores tenía una pregunta para el cliente, debía llenar el formulario de la Figura 4.10 que se encontraba en un documento. Ese formulario era copiado y enviado por mail al líder y éste se lo enviaba a una persona del cliente. Esta persona se encargaba de reenviar la duda a quien correspondiera y una vez que estaba resuelta, era quien respondía al pedido del líder. Cuando la respuesta llegaba al líder, era enviada a quien correspondía dentro del grupo. El receptor de la respuesta debía:

- Analizar la respuesta: En caso de ser una respuesta que resolvía el problema planteado, entonces completaba el formulario que se muestra en la Figura 4.10. De no ser satisfactoria, completaba el formulario y

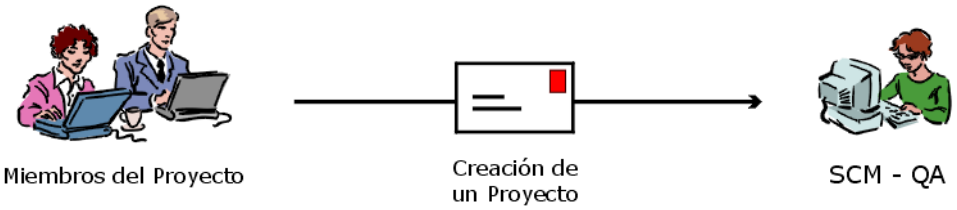

Figura 4.8: Creación de un proyecto.

Code Project Creation Request Form
Jira Link
<Jira Link>
Eclipse Project
<Eclipse Project Desired Path in CVS>
Description
<Description of purpose of Project>
Additional Comments
<Additional instructions to the configuration manager or QA, if applicable. Applicable comments are known problems, etc>

Figura 4.9: Formulario de la creación de un proyecto.

anexaba otra pregunta y retomaba el circuito. Este caso puede verse en la Figura 4.11 donde se muestra la posibilidad del ingreso de una iteración.

- Guardar la respuesta: Una vez que se completaba el formulario, si se había dado por satisfactoria la respuesta, el formulario se sacaba del documento original y se lo copiaba en el documento de dudas resueltas. De no ser satisfactoria, se mantenía en el documento original. Luego ambos documentos eran guardados en un repositorio CVS [8].

Esta forma de resolver las comunicaciones es mediante un documento de escritura colaborativa y como las conversaciones forman parte del contenido del documento las conversaciones se encontraban implícitas en el mismo.

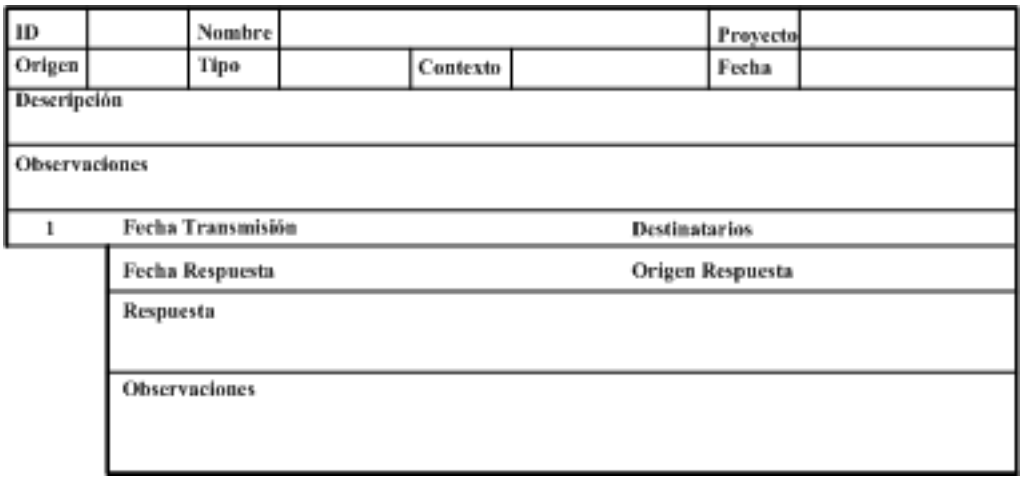

Figura 4.10: Formulario de consulta.

4.3.3 Tercer y definitivo cambio en las conversaciones

Con el paso del tiempo, y la presión de las fechas de entrega, el circuito establecido se transformó en un cuello de botella. El líder tenía muchas responsabilidades y no era eficiente la comunicación establecida como herramienta de análisis, por lo que quienes generaban las consultas, terminaban presionando al líder por una respuesta y éste al cliente. Con tantas cosas pendientes que había en el proyecto, no se podía tampoco hacer un seguimiento y exigir las respuestas a las consultas, por lo que otro esquema fue necesario.

Forma de comunicación establecida: Una persona se agregó medio tiempo al grupo y era la encargada de administrar los documentos, la comunicación con el contacto en el cliente y los tiempos de respuesta. Los documentos seguían estando disponibles pero la comunicación, con el cliente y el grupo era puramente por mail.

4.3.4 Conclusiones

La Organización para dialogar con el cliente es el mejor ejemplo de que las conversaciones por mail son efectivas pero insuficientes. La necesidad de que el conocimiento lo maneje una persona pareció ser la mejor solución para ese tipo de problemas y se terminó delegando en ella todo el conocimiento de las conversaciones en una única persona.

Esta descisión, trajo como consecuencia importantísimas restricciones:

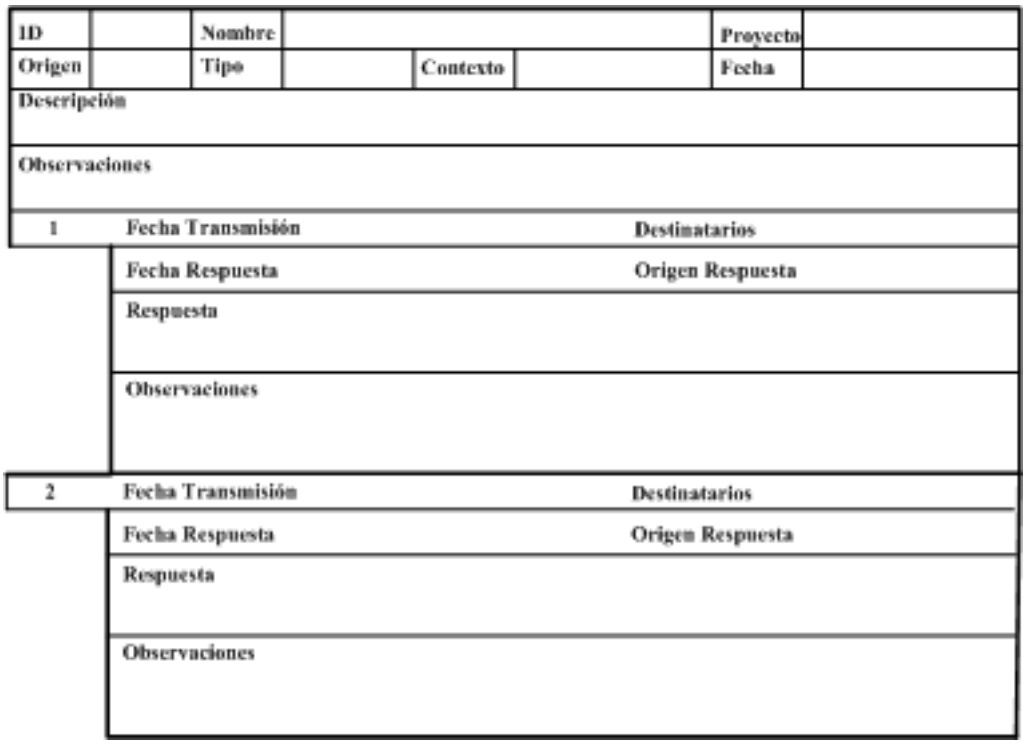

Figura 4.11: Formulario de consulta con una iteración.

- El conocimiento de las respuestas de las dudas lo tiene la persona que tiene el rol de hablar con el cliente.
- El conocimiento de las razones de demoras de respuestas o de dudas pendientes lo tiene la persona con el rol de hablar con el cliente.
- El conocimiento de esta persona es difícil de transferir.
- La historia de las conversaciones, puede ser obtenida por los documentos resultantes, e indagar sobre los mismos puede resultar inviable.
- No se logró un trabajo en equipo para solucionar e informar las comunicaciones ni se solucionaron aspectos de transmisión de experiencias.
- No se logró un trabajo en equipo para resolver la comunciación ni para tranmitir experiencia dentro del grupo de trabajo ni para otros grupos.

Capítulo 5

Diseño

El siguiente capítulo presenta la solución a la que se concluyó después de la investigación y el análisis que se reflejan en los capítulos anteriores.

Hasta el momento, se hizo referencia a la necesidad de estructurar conversaciones para los proyectos de desarrollo de software en proyectos de desarrollos inmaduros. Esta necesidad surge de los conflictos provenientes de la comunicación en estos proyectos y que fueron presentados en los capítulos anteriores.

Se mostró como las comunicaciones en los proyectos eran mantenidas con mails o documentos compartidos y sin vinculación entre las distintas herramientas lo que producía un desfasaje entre ambas, a la vez que la información no llegaba a quienes correspondía ser notificado de los temas tratados.

Dos problemas fundamentales pueden listarse brevemente que son la base de los problemas de estos proyectos:

- Las conversaciones no son miradas ni tratadas como tal, sino que son vistas como mensajes individuales. La falta de vinculación de estos mensajes, trae a consecuencia la imposibilidad de analizar y estudiar el funcionamiento de un grupo de trabajo o de una actividad.

- La falta de centralización de estos mensajes, provoca la incompletitud de las conversaciones y como consecuencia de esto la falta de mecanismos de análisis. Así existen carencias de conocimiento de ciertos aspectos del funcionamiento de un grupo de trabajo y de la posibilidad de transmisión de experiencias.

Uno de los objetivos a cumplir es que las conversaciones dentro de un proyecto de software sean vistas como una importante fuente de conocimiento

de experiencias dentro de un grupo de trabajo. Basándose en esto, se debería pensar a las conversaciones como parte esencial a organizar dentro de un proyecto de software en vías de maduración y a través de las mismas poder ver y aprovechar el trabajo diario como la fuente de experiencia del grupo

5.1 El modelo de objetos

5.1.1 Las conversaciones

Las conversaciones son el elemento de trabajo. Son el fundamento de todo el análisis hecho hasta ahora y la base para los objetivos buscados. Es el elemento interesante a capturar para entender las comunicaciones que ocurren día a día dentro del proyecto. Es el objeto protagonista de la mayoría de los requerimientos y por esta razón se describirá en detalle en esta sección.

Los problemas detallados en la Sección 3.2 sobre el uso de la voz y el video proveen fundamentos suficientes para buscar una alternativa a estas formas de comunicación, por lo que el resto de las formas de comunicarse a través de una computadora son las que se intentarán contemplar como posibles soluciones.

En las formas de comunicación a través del mail, grupos y chat la unidad de comunicación son los mensajes. En cambio en la escritura colaborativa y el workflow la forma de comunicación es el texto escrito, sin existir el concepto de mensaje. A pesar de esto último y con el objetivo de hacer lo mas general posible la solución que se está buscando, se unificará a todas los mecanismos de comunicación al término de mensaje. Hechas estas aclaraciones se diseñarán las comunicaciones en forma abstracta a las herramientas presentadas en el Capítulo 3.

Una conversación es iniciada en un grupo de personas con un objetivo. En particular, una persona comienza la misma mediante un mensaje y el resto de los participantes de la conversación responden ese mensaje.

Existen casos donde el objetivo de la conversación requiere que la respuesta al mismo sea con cierta urgencia, ya sea por ser crítica para la persona que inició la conversación o porque de alguna forma su trabajo actual depende de esa respuesta. Por estas razones, entre otras, podría esperarse que las conversaciones cuenten con una fecha límite de respuesta y a partir de esta fecha, poder controlar las conversaciones que están retrasadas o aquellas que aún no fueron respondidas.

En la Figura 5.1 se muestra un diagrama de objetos donde puede verse la relación entre las personas, los mensajes y las conversaciones.

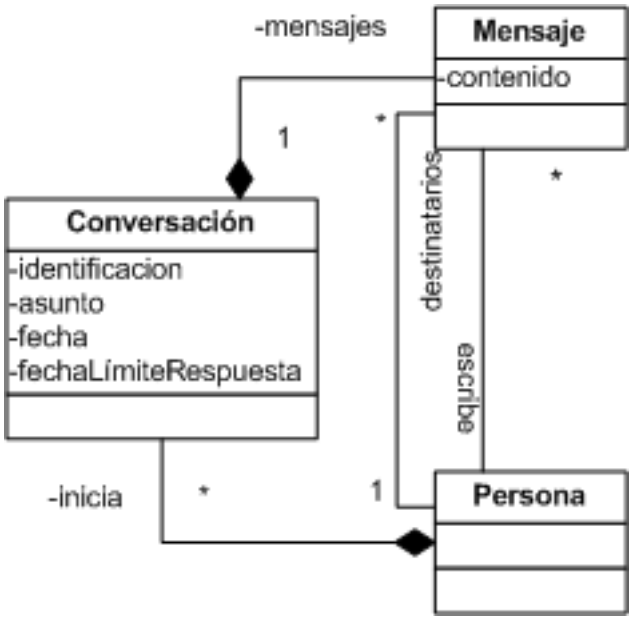

Figura 5.1: Modelo inicial.

Contando con una idea simple y básica como la mostrada recién se cubrieron los siguientes requerimientos:

- R01:Administración de usuarios.
- R04:Conversación como unidad de trabajo.

5.1.2 Los roles en un proyecto de desarrollo

Como se planteó inicialmente, la empresa u organización de software sobre la que se está trabajando tiene intenciones de mejorar la forma de trabajo. En particular, hemos hablado de CMM [5] como modelo de mejora al cual aspiran los dos grupos del caso de estudio.

CMM [5] fue desarrollado por el SEI [30] y es un framework que describe los elementos claves de un proceso de software efectivo. Marca los pasos a seguir para que las empresas inmaduras y sin organización definida para el desarrollo de software, adopten medidas de estructuración en los proyectos y puedan escalar dentro de los cinco (5) niveles que propone el modelo. De esta forma se parte de un primer nivel llamado Nivel 1 o Inicial desde donde se asume que están todas las empresas u organizaciones en un primer momento. Se sigue por el Nivel 2 o Repetible sobre el cual se comienzan a delinear los primeros pasos a seguir para lograr una organización básica

que pueda repetirse a lo largo del tiempo. Cuando los pasos del Nivel 2 se asumen como incorporados y logrados, se comienzan a adoptar los pasos del Nivel 3 o Definido. Una vez alcanzado el Nivel 3, se adoptan los del Nivel 4 o Manejado y una vez logrado este último, se adoptan las prácticas del Nivel 5 u Optimizado.

Cada Nivel en CMM [5], exceptuando el Nivel 1 - Inicial, se descompone en una serie de áreas clave del proceso (en inglés Key Process Areas) que son documentadas y archivadas. Estas áreas claves identifica un grupo de actividades relacionadas que cuando son realizadas en conjunto alcanzan un grupo de objetivos importantes para escalar en los niveles.

En el Nivel 2 es desde donde comienzan su trabajo de maduración las organizaciones, ya que se asume que todas las organizaciones ya se encuentran en un nivel Inicial. En este nivel, se definen dos áreas clave de importancia para este trabajo (cabe aclarar que no son las únicas). Estas áreas se llaman "Planeamiento de Proyecto de Software" (o SPP por el inglés Software Project Planning) y "Administración de Configuración de Software" (o SCM por el inglés Software Configuration Management). En la primera de ambas se planea, en base a estimaciones, las tareas a llevar a cabo a lo largo del proyecto. En la segunda se establece como mantener la integridad del producto a lo largo del ciclo de vida del proyecto, además de definir todas las herramientas y actividades que complementan el desarrollo de software.

Tanto en SCM como en SPP se necesitan definir qué miembros del grupo van a llevar a cabo qué tarea a lo largo del proyecto. A esta asignación se la llama rol y según el libro "The Capability Maturity Model" [37] es una unidad de responsabilidades definidas que pueden ser asumidas por uno o mas individuos. Es decir que en las primeras actividades que plantea CMM [5] se encuentra la definición de roles y la asignación de todo el trabajo en base a esos roles definidos. Es la base sobre la que se trabaja.

En particular, el funcionamiento descrito en la Sección 4.2 y en la Sección 4.3 estaban especificados dentro del documento de SCM. En cambio, en el caso de estudio de las conversaciones con el cliente, esto no había sido definido en el comienzo del proyecto en ninguna de la documentación del mismo, por lo que el rol de comunicador no había sido resuelto hasta que surgió la necesidad de agregarlo.

Entonces el modelo planteado anteriormente debe incluir no solo las personas involucradas en el proyecto, sino también los roles definidos en el mismo.

En la Figura 5.2 se introducen nuevos objetos, como son los Roles y los Proyectos. Ahora, al igual que en los documentos de un proyecto de software,

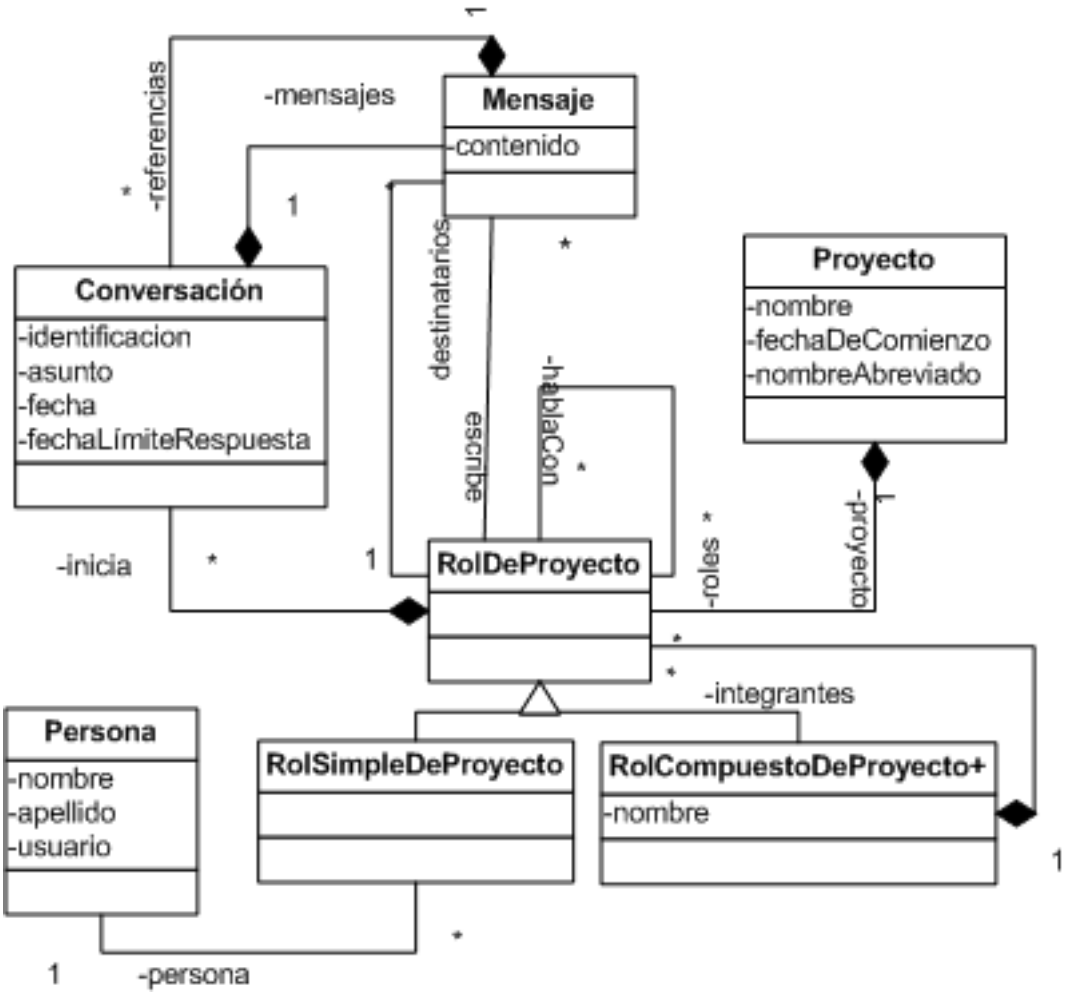

Figura 5.2: Modelo con roles.

el papel de los integrantes del grupo de trabajo, se centran en los roles. Así para un proyecto particular, se pueden definir los roles simples como: eva, juana y simona que tienen una correspondencia con las personas Eva, Juana y Simona respectivamente. También se podría definir el rol compuesto desarrollo que haría una composición de los roles eva, juana y simona. Quedan entonces cubiertos con este modelo los requerimientos:

- R02:Mecanismos de abstracción de usuarios: quedan abstraídos con los roles, tanto los simples como los compuestos.
- R09:Conversaciones grupales: mediante las comunicaciones de roles compuestos.

Definir roles dentro de la estructuración de conversaciones ayuda principalmente a desacoplar roles de personas particulares, por lo que los cambios dentro de la organización interna del grupo de trabajo no afecta ni al grupo mismo ni al cliente. Además, y como algo no menos importante, logra mantener una correspondencia entre el estado actual del funcionamiento del proyecto con las personas que trabajan actualmente. Uno de los problemas detectados con frecuencia a lo largo de la investigación, fueron los casos donde al cliente se le daba el contacto de alguno de los desarrolladores. El cliente, lejos de ser malintencionado, ante el surgimiento de algunos problemas técnicos, contactaba a este desarrollador. Como consecuencia de esto se detectaron los siguientes inconvenientes:

- Desorden en el grupo de trabajo: en muchos casos, el desarrollador que había sido consultado dejaba su tarea actual para solucionar el problema en cuestión.
- Alteración de roles: muchas veces de esas conversaciones, se tomaban decisioones que no solo no eran documentadas y comunicadas, sino que no eran tomadas por quien cumplía el rol de tomarlas, definirlas y negociarlas.
- Conflictos de liderazgos: en ciertos momentos, se perdió el foco de quien organizaba el grupo. En particular se detectó con los desarrolladores que por primera vez trabajaban en un proyecto, los cuales no sabían si responderle al líder o al cliente.
- Molestias en general: en uno de los proyectos, se detectó que durante los siguientes tres meses después que se terminó el proyecto, el cliente seguía consultando a un desarrollador por cuestiones técnicas.

Para el caso de estructurar y organizar las conversaciones en un proyecto de software, es necesario definir ciertas reglas que definan esa organización. El requerimiento asociado a este aspecto es R05:Restricción de comunicaciones y como se ve en la Figura 5.2, existe una definición en los roles que determina que rol puede hablar con cuál otro.

5.1.3 Contextualización de conversaciones

Actualmente, en herramientas colaborativas como el chat, entre otras, se pueden encontrar ciertos mecanismos que pueden verse como contextualización. Ejemplo de esto son la posibilidad de enviar símbolos que no solo realzan las expresiones emocionales, sino que también logran contextualizar al lector del

mensaje en el sentimiento de quien escribe. Esto se logra mediante símbolos que se traducen en caritas o sonidos.

Si bien para el modelo que se está buscando proveer soporte a estos mecanismos de contextualización emocional en los mensajes resulta un detalle mínimo, no puede pasarse por alto el análisis y la búsqueda de otras formas de contextualización.

Dentro de las conversaciones de un proyecto de software, pueden haber ciertos ornamentos que complementen las conversaciones y las contextualicen. Como ejemplo de esto, puede citarse los templates de mail que se presentó en el caso de estudio del repositorio de código.

Una forma de contextualizar una conversación en un proyecto de software es cuando se habla de un fragmento de código o de alguna sección de documentación. Cuando se está consultando por como se trabaja con algún tipo de tecnología o se tiene alguna duda puntual de desarrollo, se puede hacer referencia a una sección de código por ejemplo, por lo tanto el sistema de mensajería debería soportar la vinculación del nombre del archivo la que hace referencia con el archivo físico. Para esto se presentan dos casos. El primero es cuando se necesita hacer referencia a una única versión de un documento. En este caso, el documento debería mantenerse en la versión a la que se está haciendo referencia en el momento. El segundo caso es cuando se hace una referencia a la última versión disponible del documento, por lo que en todo momento debería estar actualizado el vínculo a la última versión del mismo.

Otras de las formas posibles de contextualizar una conversación es invocando a lo conversado en otras ocasiones. Por ejemplo en los correos electrónicos no existe otra forma de referenciar a otro mensaje que no sea adjuntándolo al mensaje que se está enviando o cortando y pegando fragmentos de la conversación relacionada. En el primero de los casos, si se sigue respondiendo ese mail con la conversación adjunta, el mensaje adjunto deja de formar parte de lo que se está hablando, cuando tal vez el contexto de lo que se está hablando hace que ese mensaje sea necesario en la conversación. En ambos casos, se torna difícil deducir las conversaciones vinculadas justamente por el hecho de que hay que encontrarla con alguna búsqueda avanzada de un cliente de mail. En algunos casos, se pueden tomar decisioones cuyas justificaciones y análisis habían sido conversadas previamente y estas conversaciones pueden pueden ser parte de los fundamentos de esas decisioones. Actualmente la posibilidad de poder recopilar esa historia, si es conversada mediante el correo electrónico, depende de los integrantes de la organización.

En la Figura 5.3 se presenta el diseño de objetos que resuelve la con-

textualización buscada. En la figura se muestran cómo los mensajes pueden vincularse con archivos en las dos formas anteriormente planteadas. Además, se muestra como los mensajes pueden vincularse con otras conversaciones mediante su referencias.

5.1.4 La publicación de las conversaciones

Existen ciertas conversaciones que, como se explicaba anteriormente, proveen los fundamentos de la toma de decisioones. Si por ejemplo en una conversación surge un problema particular de una persona, como puede ser la dificultad del uso de una herramienta, el debate por solucionar el problema y la conclusión a la que se arriba puede ser de importancia para otros proyectos así como también para otros miembros del mismo proyecto con dificultades similares.

Inicialmente se pueden detectar al menos los siguientes casos particulares donde las conversaciones deberían hacerse públicas para aprender de las conversaciones:

- Toma de decisiones: cuando alguna decisión dentro del proyecto se toma a través de una conversación o debate entre un grupo de gente. Esto provoca que estén disponibles los fundamentos y las conversaciones en torno a esa decisión pudiendo ver por qué se tomaron las mismas.

- Formas de trabajo: hay conversaciones de intercambio de cómo trabajan los distintos miembros de un grupo de desarrollo. Un ejemplo detectado en uno de los casos de uso, fueron los pasos a seguir para la configuración y uso del repositorio de código.

- Dudas técnicas: en el caso de estudio de la conversación con el cliente, en el momento donde parte del grupo de desarrollo estuvo en el cliente, se detectaron muchas conversaciones de forma de resolución de problemas técnicos. Muchas de esas conversaciones hacían referencia a cómo resolverlas y la publicación de las mismas hubieran sentado precedente para siguientes dudas similares, así como también para el análisis de aspectos de capacitación de futuros desarrolladores.

- Buenas prácticas: dentro del grupo pueden detectarse buenas formas de trabajo que pueden surgir a partir de conversaciones o simplemente pueden comunicarse y publicarse.

- Referencias desde documentos como minutas de reunión o planes de proyectos.

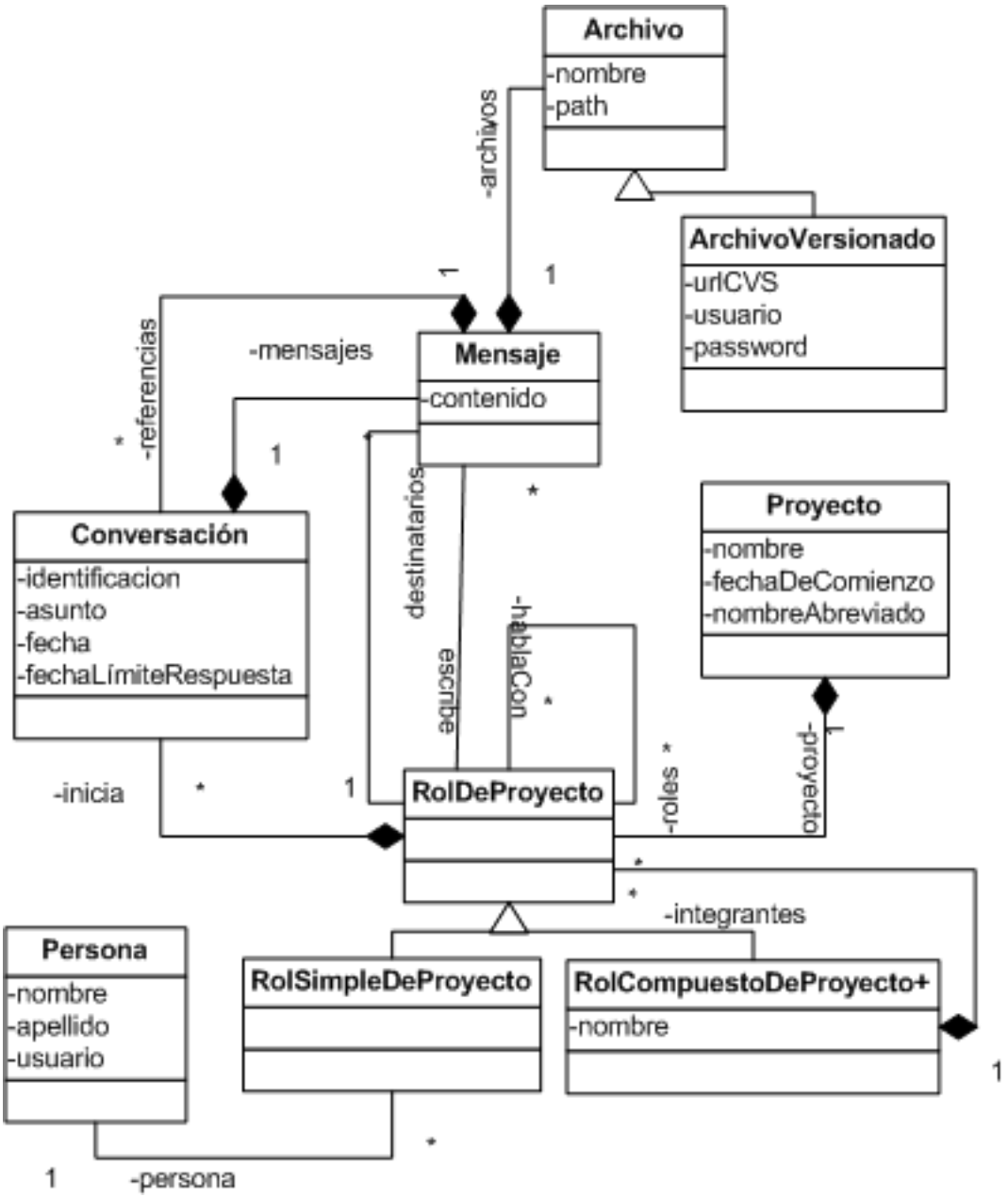

Figura 5.3: Modelo con archivos.

Estas publicaciones, pueden ser relevantes para distintas personas que posiblemente no participaron de la conversación, por lo que la publicación de las mismas podrían ser notificadas a quienes expresen interés de estar al tanto de ellas.

5.1.5 Notas sobre las conversaciones

La finalización de un proyecto de software sienta precedentes para otros proyectos. Los esfuerzos durante el desarrollo, la experiencia adquirida y progreso del grupo sirven como fundamentos para futuros proyectos. Es importante por eso hacer un análisis post mórtem del mismo.

Cuando se piensan en los análisis post mórtem, se busca sacar conclusiones de las acciones tomadas a lo largo del proyecto. Las conversaciones, son entonces una buena fuente de información para realizar ese balance y el mismo puede contradecir ciertas decisioones u opiniones de etapas previas del proyecto. Estas contradicciones complementan a las conversaciones y permiten archivar fehacientemente un proyecto.

Para complementar las conversaciones, sería deseable poder hacer anotaciones sobre las mismas para poder adjuntar conclusiones tiempo después de si las decisioones fueron o no correctas.

5.1.6 Diseño final

En la Figura 5.4 se presenta el modelo completo del diseño planteado. En él, se completa el protocolo del objeto conversación según lo especificado en la sección anterior y también el diseño de categorías e intereses de las personas sobre las mismas.

5.2 Diseño de la aplicación

En este capítulo se presenta la selección de tecnologías e implementación del diseño del capítulo anterior.

5.2.1 La tecnología

Muchos factores inciden sobre la elección de tecnologías en una aplicación groupware. El mayor desafío de una aplicación colaborativa es que se espera ser usada por un grupo de personas y, por tal motivo, el rechazo de al menos una persona puede provocar el fracaso de dicha herramienta.

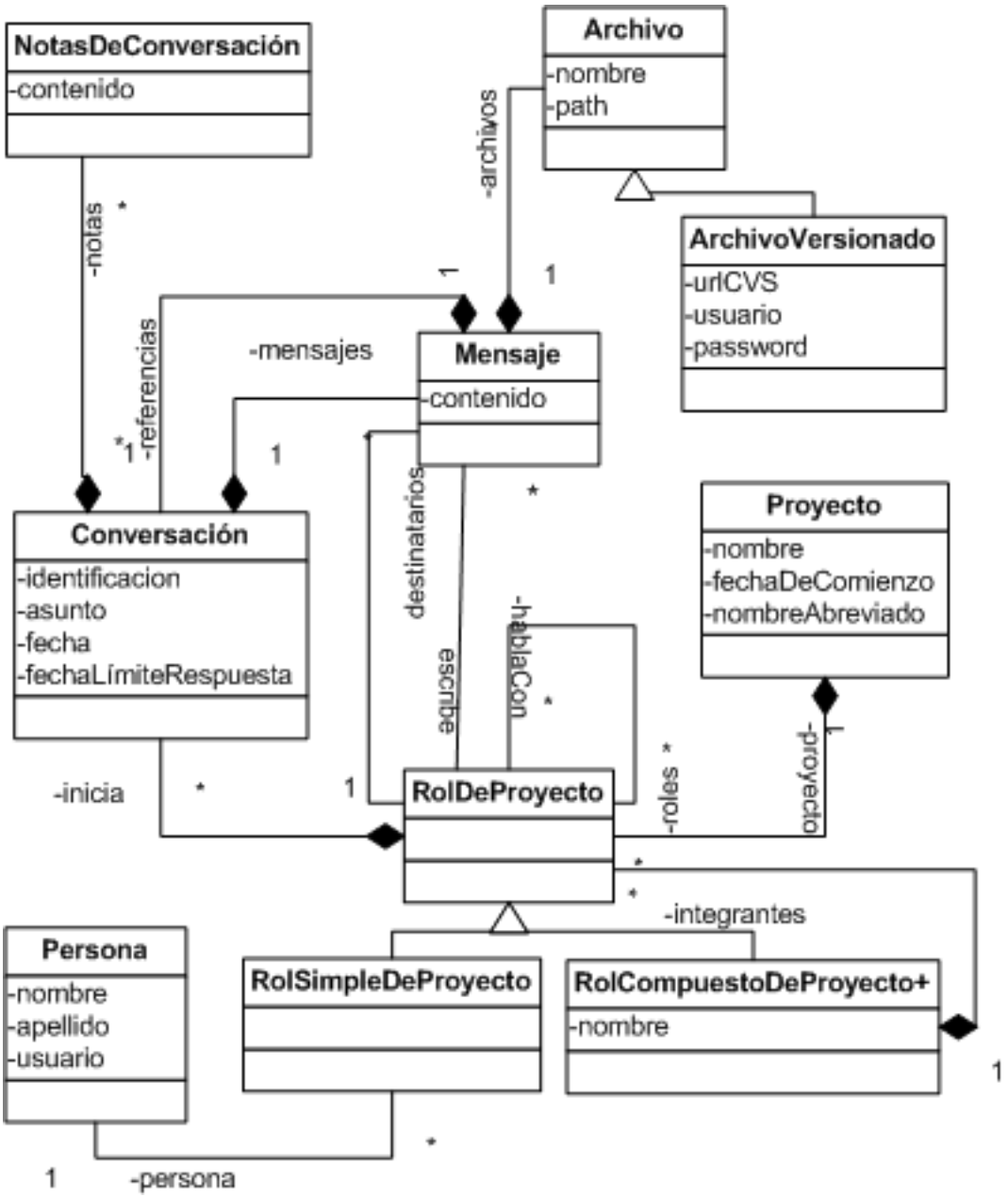

Figura 5.4: Modelo Final.

La adopción y aceptación de parte del grupo de personas que van a usar una herramienta colaborativa es lo que va a determinar el éxito de la misma. Tanto en la publicación [13] de Jonathan Grudin como en [29] de Thomas Schäl se examinan distintas herramientas buscando analizar las mismas desde el punto de vista de la dinámica de la gente que las usa. Citando un problema de cada autor, se destacan:

- La disparidad entre quién tiene que hacer el trabajo extra que la aplicación requiere y quién obtiene el beneficio de ese trabajo extra.
- La necesidad de una fuerte capacitación para aprender la herramienta.

Estos dos problemas, son la motivación para la elección de la tecnología a usar. Minimizar el impacto del uso de la aplicación y hacerlo inmediatamente aplicable y fácil de usar son los objetivos próximos.

Sin considerar la voz y el video, ya descartados desde el comienzo, se analizarán las fortalezas y debilidades de las tecnologías presentadas en el capítulo 3 para implementar entonces el diseño definido.

Sistemas de escritura colaborativa y Workflows

Cuando se relataba la estructuración de conversaciones con el cliente, el sistema de escritura colaborativa que se había elegido era un documento de texto con control de versiones y disponible al alcance de todo el grupo de desarrollo. De esta forma, en un documento se encontraban los extractos de las conversaciones que en realidad eran mantenidas vía correo electrónico. Así, existía una disparidad entre la forma de comunicación y el documento, además de una dificultosa administración del mismo.

En su momento, sistemas de escrituras colaborativas como las wiki [36] había sido descartada por dos razones. La primera era una gran crítica y resistencia a tener otra herramienta adicional a las ya existentes y la segunda era que no aportaba mayores ventajas en cuanto a organización y búsqueda de información que un documento de texto.

El diseño de un workflow que representara las conversaciones con el cliente, no solo hubiera organizado en distintos estados las mismas, sino que probablemente también hubiera podido solucionar las dificultades de administración y búsqueda de información. La búsqueda de algunos de los productos disponibles que provea la creación del workflow en forma flexible para no limitar en exceso las conversaciones hubiera sido una solución adecuada. De todas formas, seguramente hubiera tenido una gran resistencia inicial a usarse por ser otra herramienta que deberían usar todos los integrantes del equipo.

El chat

El chat en su naturaleza es sincrónico y por ello es que las partes deben estar simultáneamente disponibles para conversar. La mayoría de los problemas en las comunicaciones vistas en los proyectos de los casos de estudio requieren tiempo para derivar las consultas o analizar las respuestas y por esta razón, una forma de comunicación sincrónica no es la adecuada para el escenario planteado. A esto hay que sumarle las restricciones de seguridad que la empresa tiene y que prohíbe la salida a internet de esos programas.

La elegida: el correo electrónico

Tanto las listas como los grupos se basan en el correo electrónico, por lo que serán analizados como parte de las formas posibles que puede adoptarse en el uso de los correos.

El correo electrónico es la herramienta posiblemente más importante en las comunicaciones electrónicas y mejor aceptadas y adoptadas. Su facilidad de uso y su simplicidad es lo que hace que sea usada por cualquier persona. Su forma de comunicación asincrónica permite que quien recibe un correo pueda tomarse su tiempo para responder, ajustando así la dinámica y el tiempo de las conversaciones.

El hecho de no ser sincrónico permite lo que se llama zona de reflexión. Así cuando se recibe un mail, se puede tomar un tiempo para contestarlo y la posibilidad de leer el mensaje por segunda o tercera vez permite que se interprete en forma diferente, por lo general menos emocional.

Las otras dos herramientas vinculadas con el correo electrónico son las listas y los grupos de noticias. Las listas de correo, permiten que las personas subscriptas a la misma reciban los mensajes que son enviados y así participar de una conversación general y pública. Por otro lado los grupos de noticias son subscripciones que permiten que cuando un susccripto lo requiera, bajar las últimas noticias enviadas al grupo.

Debido a la aceptación de uso que el correo electrónico tiene actualmente, estructurar conversaciones con esta tecnología sería la solución más conveniente. Los conceptos planteados durante el diseño sobre los roles, tienen una fuerte analogía con las listas de correo y los grupos de noticias lo tienen sobre la posibilidad de expresar interés sobre algún tema particular.

Estas son las razones que justifican la búsqueda de una solución sobre esta tecnología. En las siguientes secciones se presentará implementación de cómo llevarla a cabo y extenderla en los puntos que necesitan mayor cobertura para las conversaciones.

5.2.2 La implementación

La implementación de esta herramienta se hace sobre la base de un desarrollo de un servidor de mail específico para el tema. Así, se diseña un servidor que soporta a todas las características buscadas y que puede ser administrada vía una herramienta web.

Cuando un proyecto comienza, se espera tener definido los documentos de SPP y de SCM. En el primero, se espera encontrar los roles del proyecto mientras que en el segundo se espera contemplar la inclusión y administración de la herramienta que se está describiendo.

A forma de ejemplo para entender el funcionamiento propuesto, se hará referencia a un proyecto llamado LDA cuyos integrantes son:

- Gerente del proyecto: Daniel
- Líder del proyecto: Martín
- Desarrolladores: Eva, Juana y Simona
- Testing: Roque

También se hará referencia a un dominio imaginario llamado unaorg.com que simula ser el dominio de la organización en la que se está instalando la herramienta. Seguidamente se presenta la funcionalidad principal.

Configuración inicial

Primeramente se debe dar de alta al proyecto con el nombre y una abreviación de tres letras que lo identifique. Opcionalmente se darán detalles del mismo como son el cliente, la descripción y las tecnologías usadas. Por último se especificarán las personas que componen el grupo de desarrollo. Ver la Figura 5.5.

Cuando el proyecto es creado y según los integrantes definidos, se crean cuentas de correo electrónico asociadas a esas personas y para ese proyecto. Así, para el proyecto llamado LDA en los que participan (entre otros) Juana, Eva, Simona se generan automáticamente las cuentas juana.lda@unaorg, eva.lda@unaorg.com y simona.lda@unaorg.com respectivamente.

El siguiente paso después de dar de alta el proyecto, muestra la pantalla que se ve en la Figura 5.6 donde se definen los nombres y los integrantes de cada rol. En la figura se ven que fueron dados de alta los roles Gerente y Líder y que se está dando de alta el rol Desarrollador. Cuando este último rol sea dado de alta se creará un alias llamado desarrollo.lda@unaorg.com.

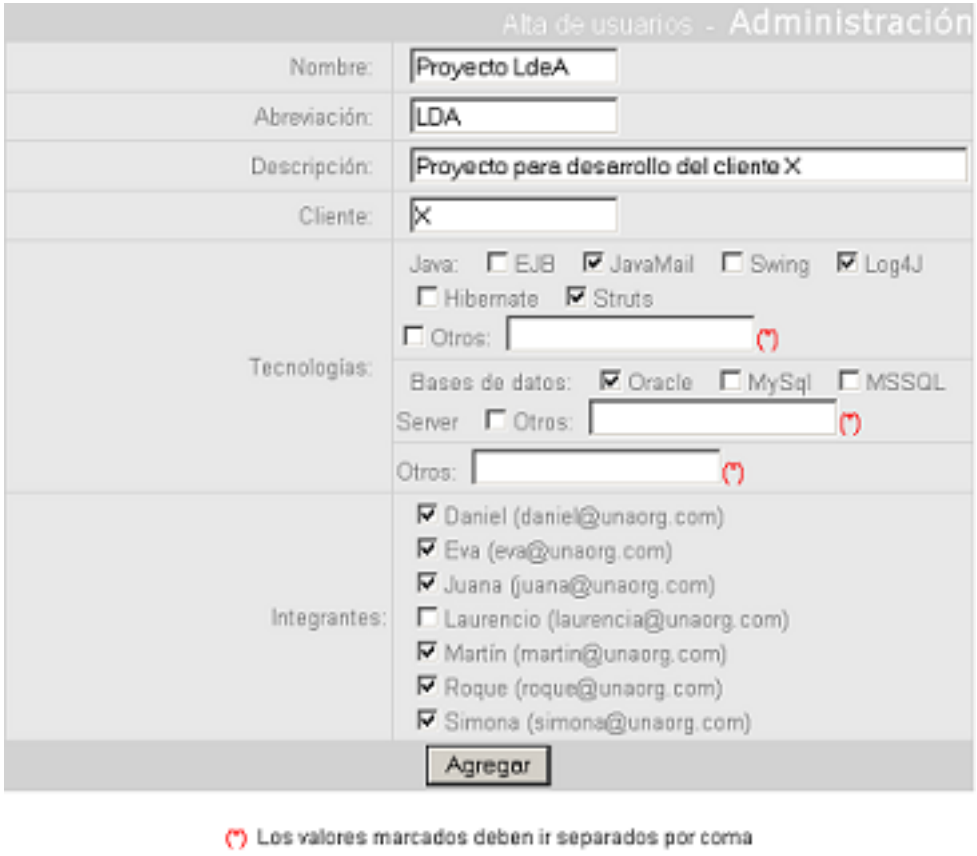

Figura 5.5: Alta de un proyecto.

Figura 5.6: Alta de los roles del proyecto.

Como tercer y último paso para el alta de un proyecto, se presenta una pantalla como la vista en la Figura 5.7 donde se definen los permisos para quien puede hablar con quien dentro del proyecto. En el ejemplo de la figura, se muestra el alta del proyecto Desarrollo que puede comunicarse con Líder y con Testing. Para poder comunicarse desarrollo con el cliente, lo deberá hacer indirectamente a partir de poder conversar con quien tiene permisos para comunicarse con el mismo.

Las conversaciones

Una vez dado de alta el proyecto y cada integrante del mismo tener configurada su cuenta de mail correspondiente, se pueden iniciar las conversaciones con el uso de esa cuenta de mail. Para esto existe una única restricción que se plantea al comienzo: una conversación debe ser dada de alta para poder iniciar el ida y vuelta de mensajes. Se proveen dos opciones, una para quienes opten por usar la herramienta web y otra para quienes usen únicamente el correo electrónico (ya sea por preferencia, simplicidad o porque no tienen acceso a internet).

En el caso de hacerlo a través de la herramienta web, se presenta la pantalla de la Figura 5.8. En la misma, se debe seleccionar el rol desde donde se

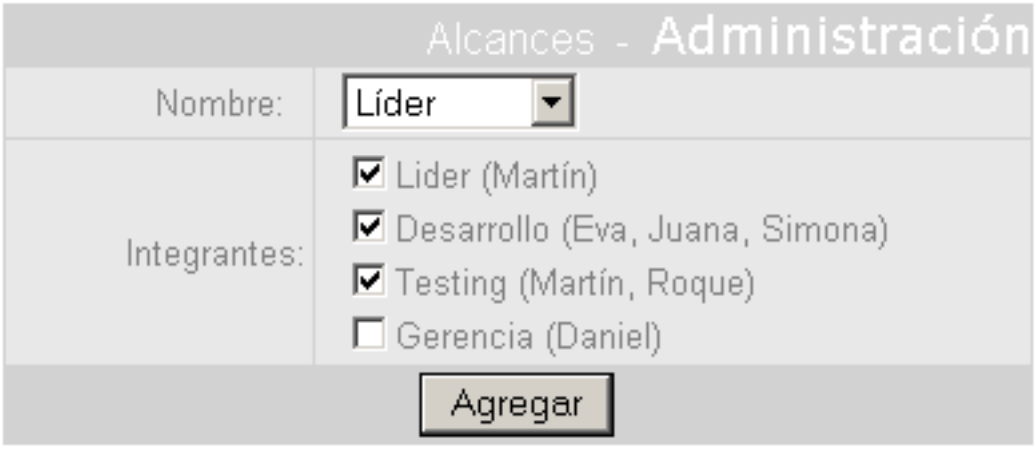

Figura 5.7: Alta de los alcances de las conversaciones.

envía el mail ya que puede existir miembros con mas de una responsabilidad dentro del proyecto. Además se debe poder seleccionar los involucrados en la conversación, quienes participan de la misma. Por último se debe ingresar el tipo de mensaje, si es crítico para el trabajo actual, la fecha de espera de respuesta y el contenido mismo del primer mensaje.

Para quienes optan por trabajar por mail, tienen dos alternativas. La primera es incorporar un template del mail de alta de las conversaciones dentro del cliente de mail y la otra alternativa es escribir un mail en blanco a la dirección nuevaConversacion para el proyecto en el que quieran conversar. Para el ejemplo que se viene desarrollando, se debería escribir a nuevaConversacion.lda@unaorg.com. En respuesta a este mail se recibirá en forma instantánea el mail de la Figura 5.9

Cuando se envía el mail de la Figura 5.9 el servidor procesa los datos ingresados, da de alta a la conversación y le envía un mensaje a cada destinatario del mail. A partir de ese momento, la conversación se mantiene haciendo "responder" a los mails recibidos.

Los mails de esa conversación, tendrán un asunto particular. Será el asunto ingresado originalmente y se verá entre corchetes el identificador de la conversación.

Las contextualizaciones de las conversaciones

Se había planteado la posibilidad de contextualizar las conversaciones con elementos que aporten información adicional a lo que se está conversando. Para esto se había planteado la necesidad de referenciar archivos del proyecto,

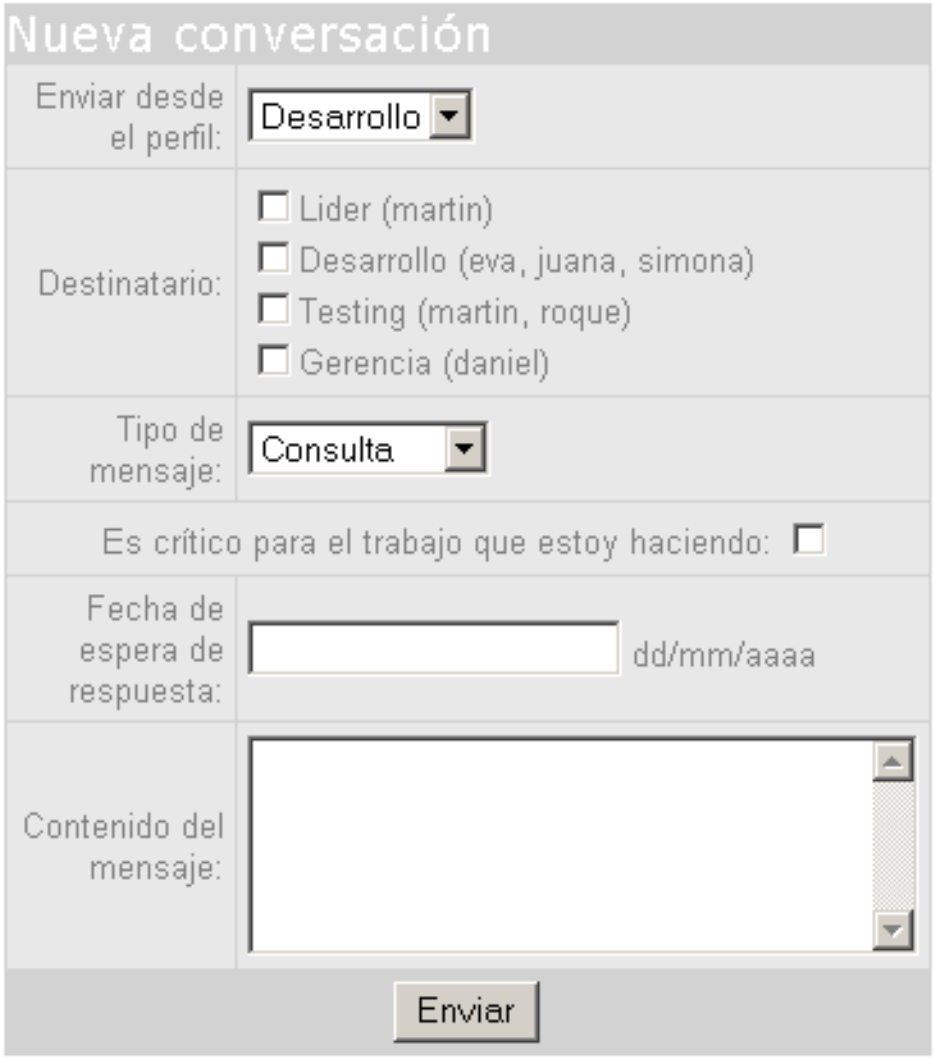

Figura 5.8: Alta de una conversación. Opción vía web.

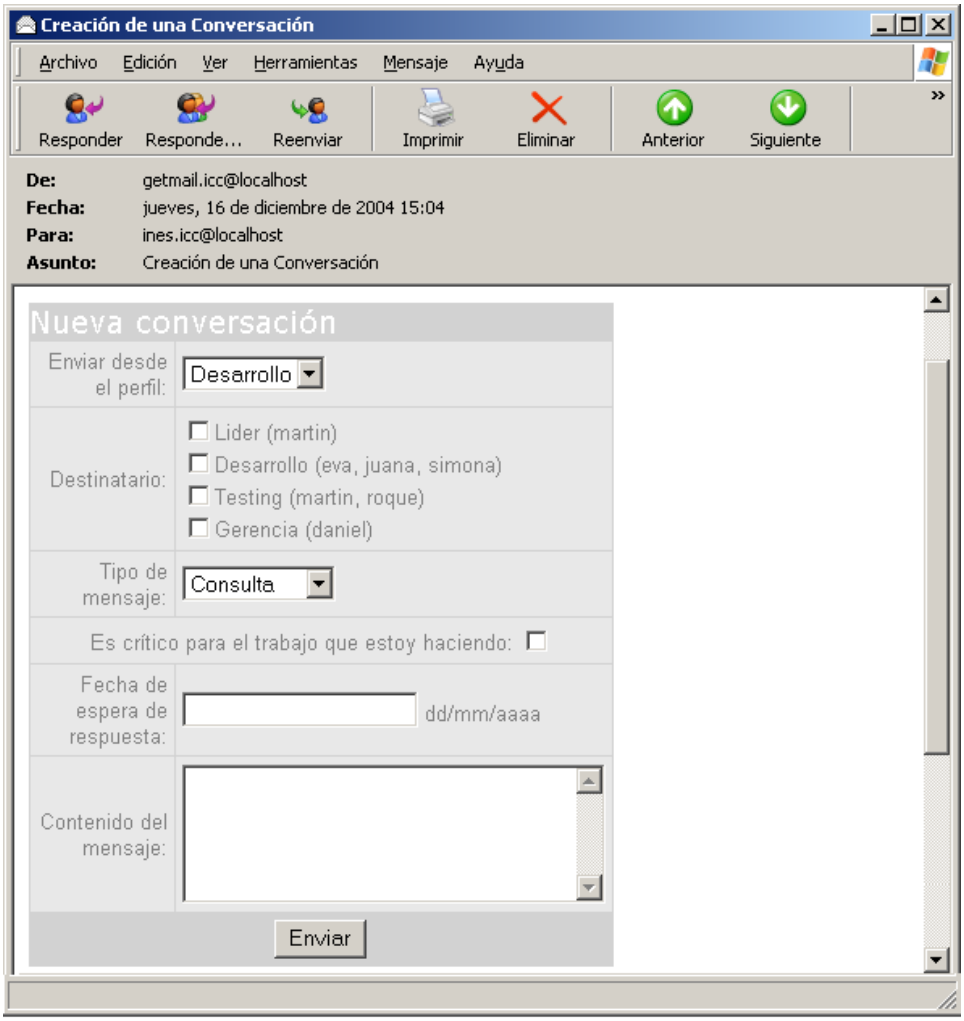

Figura 5.9: Alta de una conversación. Opción mail

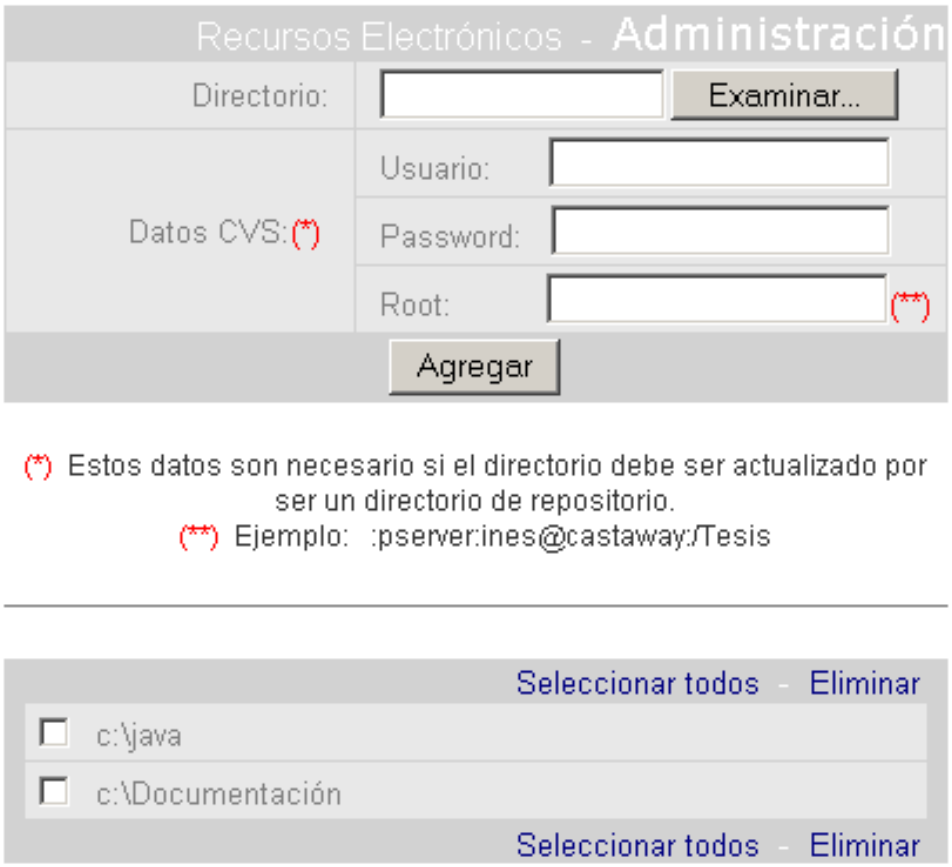

Figura 5.10: Alta de los directorios de recursos del proyecto.

así como también referencias a otras conversaciones.

La contextualización de conversaciones requiere de la configuración de los recursos del proyecto en la herramienta web. Para esto, se presenta una pantalla para darlos de alta. Ver la Figura 5.10

Para la contextualización de las conversaciones con archivos, existían dos posibilidades. Una de ellas era hacer referencia a una única versión del archivo al que se referenciaba. Esto se puede hacer incluyendo dentro del mensaje los símbolos <! y !> para acotar el nombre del archivo. Por ejemplo para referenciar el archivo Main.java, dentro del texto del mensaje se deberá escribir <!Main.java!>. Cuando el mensaje llegue al servidor, éste se encargará de buscar el archivo Main.java dentro de los recursos definidos para el proyecto y de encontrar el archivo, incluirá en el mail un link para poder recuperarlo.

Cuando se necesite hacer referencia a una versión de un documento del cual hay que buscar la última de las versiones cuando un usuario lo requiera,

se deberá escribir el nombre del archivo entre los símbolos < y >. Así si se hace referencia a Main.java, se deberá escribir <Main.java>. Cuando el mensaje con ese contenido sea recibido por el servidor, de igual forma antes descripta, buscará dentro de los recursos definidos para el proyecto y de encontrarlo, embeberá en el mail un link para recuperarlo. Cuando un usuario quiera recuperar el archivo por ese link, el servidor buscará la última versión del mismo y la enviará a quien lo requirió instantáneamente.

Así como se incluyen marcas especiales para referenciar archivos, también se podrán incluir para referenciar otras conversaciones. Así, cuando se escriba en el contenido de un mail el identificador de una conversación acotada por los símbolos [ ], se formará un link que permitirá retornar al instante la conversación referenciada en forma completa.

Las publicaciones de las conversaciones

Todos los mails de las conversaciones incluirán una barra inferior con una referencia rápida para las operaciones no cotidianas. Una de las opciones de la misma es "Publicar", que le da la instrucción al servidor de publicar esa conversación. Cuando se elige esta opción, un mail es recibido con el listado de las posibles categorías donde se puede publicarla. Las categorías pueden ser configuradas previamente desde la herramienta web o pueden ser agregadas cuando el mail de la solicitud de publicación es recibido.

5.2.3 Detalles de la herramienta complemento

La herramienta web es una herramienta que actúa como un adicional a la aplicación y su funcionamiento no compromete a la misma. La herramienta web actúa solo como una herramienta de configuración de la aplicación y es por ende un complemento.

El diseño de este trabajo, intenta resolver con el soporte del correo electrónico todas las funcionalidades por dos motivos principales. El primero es cubrir las restricciones de uso de internet que el escenario tenía y la segunda es brindar facilidad y evitar la sensación de que se cuenta con otra herramienta adicional a ser usada por el grupo de trabajo. Por este motivo, toda la funcionalidad principal se encuentra disponible a través de los mails, pero se propone la herramienta web como forma de visualización sencilla de los contenidos de la aplicación.

En la herramienta web, se encuentra toda la funcionalidad adicional para cubrir todos los requerimientos propuestos en el Capítulo 2, por lo que se

encontrará en ella todos los listados y formas de organizar la información disponible según lo planteado en los requerimientos.

Capítulo 6

Implementación

El trabajo diseñado y especificado en los capítulos anteriores fue prototipado para demostraciones de su aplicabilidad. El prototipo fue llamado PMail por Project mail y se adjunta en el trabajo. Este prototipo representa la funcionalidad mas relevante para la aplicación y fue desarrollada con tecnología lo suficientemente robusta para que su futura evolución sea la herramienta completa. Es decir que el prototipo sugiere esta tecnología para ser tenida en cuenta para el desarrollo completo de la aplicación.

En las siguientes secciones se presentará la arquitectura de las aplicaciones desarrolladas y se presentarán las razones por la que esas tecnologías fueron elejidas.

6.1 La aplicación en capas

La arquitectura de la aplicación está basado en un modelo en capas. Como se ve en la Figura 6.1, la aplicación se divide en cuatro capas bien definidas.

El objetivo de dividir en esta forma la aplicación es lograr independencia en la funcionalidad que puede estar bien sectorizada y definida. De esta forma se logra transparencia en el momento de cambiar algunas de las capas ya sea por la elección de otra tecnología o por cualquier otra razón a la vez que se logra un bajo acoplamiento que hace que las modificaciones y ajustes sean mas fáciles de hacer.

En la figura 6.2 se presenta mas detalladamente la arquitectura.

6.1.1 La capa de modelo

La capa del modelo representa el modelo en objetos de la aplicación. Es el corazón de la herramienta y en ella se encuentra toda la funcionalidad

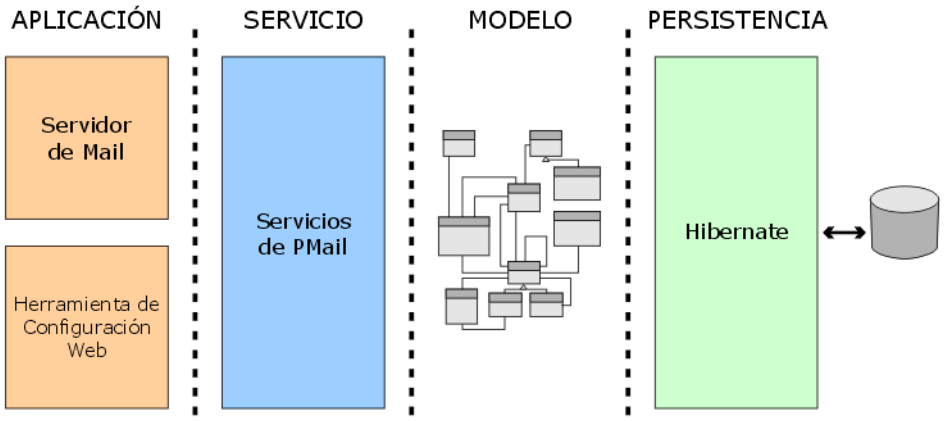

Figura 6.1: Arquitectura en capas de la aplicación.

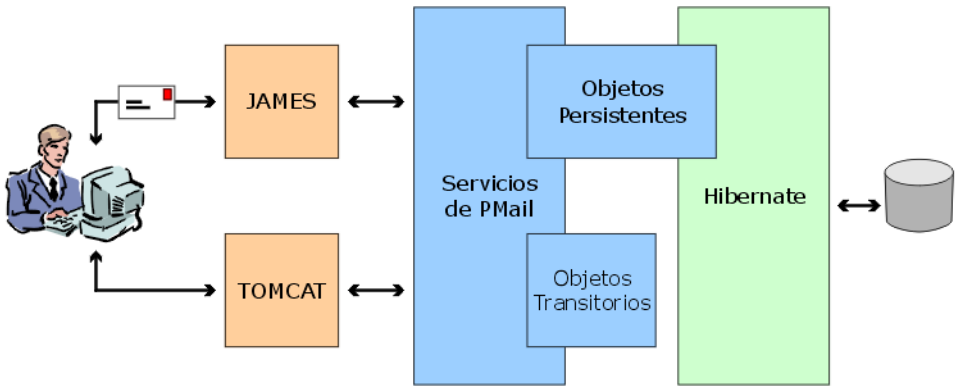

Figura 6.2: Arquitectura general de la aplicación.

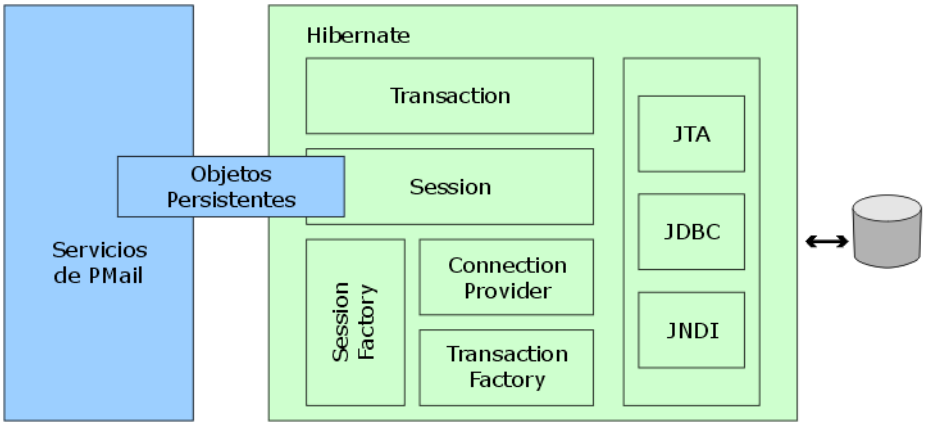

Figura 6.3: La arquitectura de la capa de persistencia.

buscada en los capítulos anteriores.

El modelo de objetos en esta capa es el presentado en la Figura 5.4 con algunas modificaciones que fueron necesarias en el momento de la implementación. Las modificaciones mas significativas fueron:

- La necesidad de tener una variable de instancia adicional que identifique a cada objeto. Esto surgió como necesidad de la capa de persistencia.
- La necesidad de proveer un constructor vacío para cada uno de los objetos. Esto también es una restricción del mecanismo de persistencia.
- La incorporación de clases necesarias para la administración de los objetos.

6.1.2 La capa de persistencia

La capa de persistencia se detalla en la Figura 6.3. La arquitectura que se presenta en la figura es la que Hibernate [14] provee por la naturaleza del diseño mismo de la herramienta de persistencia.

6.1.3 La capa de servicio

La capa de servicio está implementada usando el patrón de diseño Facade [9]. Cualquier servicio que se necesite del modelo de objetos, será requerido a esta capa que provee la abstracción necesaria para la manipulación del modelo.

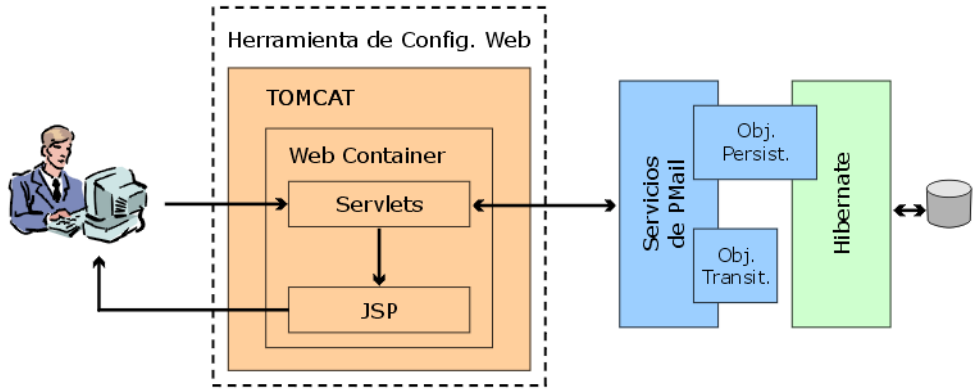

Figura 6.4: Detalle de la arquitectura de la capa web.

Usar este patrón de diseño y la independencia que significa modularizar todos los servicios en esta capa permiten se pueda diseñar en una sección bien acotada los mecanismos de persistencia. El problema de la persistencia de objetos en java es que hay que embeber en algún objeto la funcionalidad de apertura de conexión hacia la base de datos sobre la cual se persiste. Además se deben manejar las transacciones asociadas al guardar el modelo de objetos. En el prototipo implementado esa funcionalidad se encuentra en esta capa y el diseño de la misma permite que el cambio del mecanismo de persistencia se logre mediante el reemplazo de una clase particular.

6.1.4 La capa de aplicación

La aplicación es la capa que ve el usuario y con la que el mismo interactúa para hacer funcionar el prototipo. Esta capa está dividida en dos partes. Una parte es la de funcionamiento de la herramienta y la otra es de configuración de la misma. Los dos módulos están implementados independientemente y puede funcionar la aplicación de mail sin que la herramienta de configuración esté disponible.

El desarrollo de cada una de las aplicaciones fue hecho sobre frameworks ya existentes. En particular se usó para la aplicación de mails el servidor de mail de Apache [2] y para la herramienta de configuración se usó JSP y Servlets [31] y Struts [35]. Estas tecnologías serán descriptas en la siguiente sección, pero se presenta en la Figura 6.4 la arquitectura general.

La aplicación de configuración respeta el Modelo 2 de Jsp/Servlets [31] haciendo que los Servlets sean los objetos que procesan la información re-

querida por el usuario. Luego de procesar el pedido y obtener la respuesta, el Servlet le transfiere el control a la página JSP que es quien procesa la misma para presentarla visualmente. Se trabaja entonces respetando el patrón de diseño arquitectural MVC [23] (por las palabras en inglés Model View y Controller) que, explicándolo sintéticamente, propone separar el modelo que tiene la aplicación (Model) de la vista de presentación de la aplicación (View) y comunicar estas dos partes por un controlador (Controller) logrando no mezclar la lógica de la aplicación con la presentación visual.

6.2 Las tecnologías utilizadas

6.2.1 La persistencia

En la capa de persistencia se seleccionaron dos tecnologías particulares: el mecanismo de persistencia y la base de datos. El mecanismo de persistencia elegido fue un mapeador Objetos-Relacional, es decir, una herramienta que a partir de un modelo de objetos los guarda sobre una base de datos relacional. En particular se optó por Hibernate [14] por lo poderosa que es la herramienta y lo transparente que es respecto a la no invasión que tiene sobre el modelo de objetos.

Respecto a la base de datos, se eligió MySQL [27] por ser un producto de código abierto y simple de instalar y usar. Si bien esta elección fue arbitraria, al elegir Hibernate [14], y tomando los recaudos necesarios, es posible cambiando solo un archivo de propiedades y sin modificar el código de la aplicación reemplazarla por otra base de datos relacional que Hibernate [14] soporte. Es decir que hace que pueda ser configurada la herramienta con cualquiera de las mas de 18 bases de datos que soporta Hibernate [14] sin modificar el código de la aplicación.

6.2.2 La tecnología de la herramienta de configuración

La herramienta de configuración fue implementada con JSP/Servlets [31] como se contó anteriormente. La elección de esta tecnología se debió a que se buscaba tener una herramienta disponible para configurar la aplicación en forma sencilla. Al ser una aplicación Web el cliente no necesita de instalación y como se puede publicar en internet en forma sencilla y segura, también funciona para redes externas a la organización.

Además, se utilizó el framework Struts [35] para el uso de sus librerías de tags y como ayuda para respetar el Modelo 2 de las aplicaciones Web. Struts [35] es un framework de Apache [2] que propone implementar objetos subclase

de Action para representar cada una de las acciones que el cliente solicita. Por dentro el framework no hace otra cosa que forzar a que todos los pedidos lleguen a un Servlet particular el cual, implementando el patrón de diseño Command [9], direcciona al Action concreto que implementa la funcionalidad buscada. A su vez, el objeto Action concreto tiene una redirección que fuerza que el flujo continúe hacia una página JSP una vez que el pedido fue resuelto.

El prototipo presentado, se encuentra publicado en el servidor de apliaciones web desarollado por Apache [2] llamado Tomcat [32]. La elección de este servidor se debió a que es la implementación de referencia de la tecnología JSP/Servlets [31], lo que significa que una aplicación web en Java que funciona en este servidor debería funcionar en cualquier otro servidor, a menos que el mismo no respete la definición que Sun hizo sobre esa tecnología.

6.2.3 La herramienta de correo

La herramienta de correo está implementada sobre el motor de correo electrónico desarrollado por Apache llamado James [18] (por las siglas de "Apache Java Enterprise Mail Server"). James [18] es un servidor de correo y noticias basado en Java.

Este servidor provee toda la funcionalidad de un servidor de mails incluyendo los protocolos SMTP y POP3 para poder enviar y recibir mails a la vez que permite extender el framework con objetos desarrollados para alguna funcionalidad específica. Para esto provee los denominados "Matchers" y "Mailets" que interceptan los mails bajo un criterio específico y los deriva a una funcionalidad particular respectivamente.

La administración de James [18] como servidor de mails es muy sencilla y se realiza mediante una conexión telnet al puerto 4555 desde donde se pueden crear los usuarios, crear alias sobre los usuarios, hacer redirecciones, entre otras cosas.

Internamente James trabaja sobre el framework JavaMail Api [20] y toda la manipulación interna de los mails que hace el prototipo presentado trabaja utilizando esta API.

6.2.4 Las herramientas para el desarrollo y las técnicas de desarrollo

Durante el desarrollo de este prototipo se usaron varias herramientas y frameworks que ayudan a desarrollar con calidad los productos. Entre los mas importantes se detallan:

- Log4J [22]: es un framework que permite auditar lo que sucede en una aplicación en una forma muy ventajosa mediante la definición de niveles de error y de la posibilidad de configurar el nivel según las necesidades de mantenimiento. Así en una etapa de desarrollo se puede poner un nivel de detalle tal que registre el funcionamiento detallado del sistema y en una etapa de producción hacer que el nivel de detalle sea acorde a la madurez del producto (solo información relevante en un principio y errores graves cuando la herramienta se fue mejorando con los errores simples). Además permite configurar herramientas remotas, como pueden ser las de la capa de aplicación del prototipo, de forma tal que envie mails en caso de errores críticos.

- Ant [1]: es una herramienta que permite automatizar el trabajo asociado a la programación en Java. Entre otras aplicaciones, se la usó para automatizar la generación de archivos con extensión .jar .sar y para compilar y mover fuentes y archivos.

- Javadoc [19]: es una herramienta para generar documentación API en HTML. Si el código desarrollado cumple con la especificación que establece JavaDoc [19] corriendo la herramienta se genera toda la documentación que en el código fuente.

Capítulo 7

Conclusiones

7.1 Contribución

En este trabajo, se presentan los problemas de comunicaciones que frecuentemente surgen en los proyectos de desarrollo de organizaciones inmaduras. Se presentaron ejemplos de proyectos concretos sobre las consecuencias de la falta de estructruación y definición de las conversaciones. Se vieron consecuencias en los cronogramas del proyecto cuando por falta de detección de demoras en respuestas del cliente sobre consultas críticas. También se vieron consecuecias que afectaron al desarrollo del sistema ya que los cambios de especificación del mismo no eran comunicados a todos los afectados por el cambio.

Se planteó la importancia de establecer la comunicación y organizarla para que el proceso de desarrollo de software sea un trabajo en equipo. Además, se brindó una solución para que esa comunicación siente las bases y fundamentos para que el siguiente proyecto sea una evolución, en términos de maduración del grupo, en términos de maduración de la organización y en términos de transmisión de conocimiento y experiencias entre los involucrados del grupo.

Para solucionar los problemas detectados y con el soporte de un servidor de mail extendido, se presentó una propuesta para la estructuración y organización de las comunicaciones dentro de un proyecto. Como elemento central de la propuesta se introdujo el concepto de conversación planteando a las mismas como forma única de comunicación.

Contando con un servidor de mail extendido se configura las formas de comunicación y los permisos de comunicación. Como parte de una configuración incial del mismo se crean cuentas de mail específicas para cada integrante del proyecto, así como alias para los subgrupos y roles dentro del

mismo. Usando el correo electrónico como base tecnológica para estructurar las conversaciones y las funcionalidades de las listas de correo y los grupos de noticias se logró un balance de configuración para definir las comunicaciones de los proyectos. En particular se introdujeron los siguientes conceptos para la configuración de las comunicaciones en proyectos de software:

- Los mails públicos: Los mensajes forman comunicaciones y las comunicaciones sirven para la colaboración, comunicación y cooperación del grupo de desarrollo.

- Correspondencia entre listas de correo y roles dentro del proyecto: Cada subgrupo dentro del proyecto tiene su lista de mails de destinatario para conversar en grupo. El cambio de las asignaciones dentro del grupo de trabajo implica no solo un cambio en la definición de la documentación correspondiente sino también una actualización en la configuración de ese proyecto particular en la herramienta.

- Suscripción a novedades relativas al proyecto: Mediante la configuración y preferencia de cada miembro del proyecto que definen su interés sobre publicaciones o conversaciones.

La solución planteada, cubre ampliamente los objetivos buscados por Alistair Cockburn [6] en su "Manifiesto del juego cooperativo" desarrollado en la Sección 2.2.

El trabajo en su totalidad aplica, con una sutil diferencia, a los temas analizados por Adele Goldberg sobre "Collaborative Software Engineering" [11]. La diferencia radica en que en [11] Goldberg hace referencia al relato de experiencias como base de análisis mientras que en el trabajo desarrollado se plantea la conversación diaria como base de ese análisis. En una forma mas sencilla, se intenta organizar las conversaciones para que el día a día sea un relato de experiencias y así compatir el conocimiento, enseñar el uno al otro o simplemente aprender algo nuevo.

A partir de la idea de contar historias sobre experiencias, Goldberg [11] busca la transmisión de experiencias en:

- Aprender sobre la conducta del equipo: cómo trabajan en grupo, cómo responden, como participan y colaboran.

- Aprender sobre los que participan del equipo: mediante la evaluación post mórtem del proyecto y con un análisis de cada persona sobre su rol, permite transmitir a otros grupos de desarrollo posteriores experiencias.

- Aprender de la práctica: monitoreando el proceso y el flujo del conocimiento.

En este trabajo no se analiza contar historias pero permite que se transmita experiencia a partir de las conversaciones mediante el análisis que referencia Goldberg en [11].

Aprovechando los eventos del día a día pueden detectarse debilidades que podrían ser consideradas en futuras capacitaciones o fortalezas del grupo. Estos son solo dos ejemplos entre muchas de las aplicaciones que tienen y son ejemplos de análisis simple con lo que uno cuenta con la herramienta. Sin embargo, aspectos a analizar del comportamiento y la dinámica del grupo pueden no desprenderse de la interacción diaria. La solución planteada ofrece un lugar para analizar si esto sucede y en qué forma, y permite disparar actividades de debate o inquietudes para provocar situaciones que permitan este análisis.

7.2 Trabajo futuro

Si bien el trabajo fue planteado en un entorno caótico y se planteó una solución para diciplinar y ogranizar las comunicaciones en este caos, esta solución brinda fundamentos suficientemente fuertes para no ser descartada cuando la organización madura. La existencia de esta solución trasciende el objetivo de diciplinar y organizar y se vincula directamente como un asistente para la mejora de la colaboración de un grupo.

Pueden entonces extenderse la solución planteada para mejorar la organización de las conversaciones y las formas de búsqueda de tópicos particulares, incorporándole inteligencia a la misma.

Otra de las cosas a mejorar y resolver, es lo público de las conversaciones. El hecho de que las conversaciones sean públicas puede provocar cierta autocensura de los integrantes del grupo a conversar con soltura. Esto se debe al miedo de cada persona de ser juzgado por sus acciones. Como trabajo futuro, se debería analizar y resolver ese tipo de problemas en base a estudios del funcionamiento de los grupos humanos.

Un aspecto importante a ampliar y fortalecer es el debate sobre porciones de código.

Agradecimientos

Los proyectos analizados y evaluados en este trabajo pertenecen a trabajos realizados dentro del LIFIA - Laboratorio de Investigación y Formación en Informática Avanzada de la Universidad Nacional de La Plata. Las opiniones expresadas en el mismo no intentan representar la visión de ninguno de los miembros del LIFIA, siendo estas opiniones personales de quien escribe el trabajo.

Bibliografía

[1] Ant. *http://ant.apache.org/* Último acceso: 19 de Mayo de 2005

[2] Apache Software Foundation. *http://www.apache.org/* Último acceso: 19 de Mayo de 2005

[3] Mozilla.org's Bugtracking System. *http://bugzilla.mozilla.org/* Último acceso: 19 de Mayo de 2005

[4] Clarence A. Ellis and Simon J. Gibbs and Gail Rein , 1991. *Groupware: some issues and experiences.* Communications of the ACM. Volume 34 , Issue 1. Pp. 39-58

[5] Capability Maturity Model. *http://www.sei.cmu.edu/cmm/* Último acceso: 19 de Mayo de 2005

[6] Alistair Cockburn. *http://alistair.cockburn.us/* Último acceso: 19 de Mayo de 2005

[7] Wolfram Conen. Gustaf Neumann (Eds.) 1997. *Coordination Technology for Collaborative Applications.* Springer

[8] Concurrent Versions System. *http://www.cvsnt.com/* Último acceso: 19 de Mayo de 2005

[9] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides , 1995. *Design Patterns..* Addison-Wesley Publishing Company. ISBN 0-201-6361-2

[10] GMail. *http://www.gmail.com/* Último acceso: 19 de Mayo de 2005

[11] Adele Goldberg: *Collaboratory Software Engineering.* Publicado en: Net Object Days, Erfurt, Germany, Oct 12,2000.

[12] Google. *http://www.google.com/* Último acceso: 19 de Mayo de 2005

[13] Jonathan Grudin, 1998. *Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces.* CSCW '88, Proceedings of the conference on Computer-supported cooperative work. ACM Press. New York. Pp. 85-93.

[14] Hibernate: Relational Persistence For Idiomatic Java. *http://www.hibernate.org/* Último acceso: 19 de Mayo de 2005

[15] Hotmail Grupos. *http://groups.msn.com/* Último acceso: 19 de Mayo de 2005

[16] Watts S. Humphrey: *Managing the Software Process.* SEI Series in Software Engineering. ISBN:0-201-18095-2

[17] ICQ. *http://www.icq.com/* Último acceso: 19 de Mayo de 2005

[18] James. *http://james.apache.org.* Último acceso: 19 de Mayo de 2005

[19] Javadoc Tool. *http://java.sun.com/j2se/javadoc.* Último acceso: 19 de Mayo de 2005

[20] Java Mail API. *http://java.sun.com/products/javamail/* Último acceso: 19 de Mayo de 2005

[21] JIRA. *http://www.atlassian.com/software/jira/* Último acceso: 19 de Mayo de 2005

[22] Log4J Logging Service. *http://logging.apache.org/log4j/* Último acceso: 19 de Mayo de 2005

[23] Glen E Krasner and Stephen T.Pope. *A cookbook for using the model-view controller user interface paradigm in Smalltalk-80.* Journal of Object Oriented Programming, 1, (3):26-49, August / September 1988.

[24] Mantis Bugtracking System. *http://mantisbt.sourceforge.net/* Último acceso: 19 de Mayo de 2005

[25] MSN Messenger. *http://messenger.msn.es/* Último acceso: 19 de Mayo de 2005

[26] Microsoft Word. *http://office.microsoft.com.* Último acceso: 19 de Mayo de 2005

[27] My SQL. *http://dev.mysql.com/* Último acceso: 19 de Mayo de 2005

[28] Open Office. *http://es.openoffice.org/* Último acceso: 19 de Mayo de 2005

[29] Thomas Schäl, 1996. *System Design for Cooperative Work in the Language/Action Perspective*. In Shapiro, D.; Tauber, M.J.; Traunmueller, R. (eds). The Design of Computer Supported Cooprative Work and Groupware Systems. Elsevier Science, Amsterdam: 377-399

[30] Software Engineering Institute - Carnegie Mellon University. *http://www.sei.cmu.edu/* Último acceso: 19 de Mayo de 2005

[31] Java Servlet Technology. *http://java.sun.com/products/servlet/* Último acceso: 19 de Mayo de 2005 y *http://java.sun.com/products/jsp/* Último acceso: 19 de Mayo de 2005

[32] Tomcat. *http://jakarta.apache.org/tomcat/* Último acceso: 19 de Mayo de 2005

[33] D.Shapiro. M.J.Tauber. R.Traunmüler. 1996. *The Design of Computer Supported Cooperative Work and Groupware Systems*. North - Holland

[34] *Complex Acts of Knowing: Paradox and Descriptive Self-Awareness*. Publicado en: PROCEEDINGS of the THE 3rd EUROPEAN CONFERENCE ON KNOWLEDGE MANAGEMENT Trinity College Dublin, Ireland 24-25 September 2002

[35] Struts. *http://struts.apache.org/* Último acceso: 19 de Mayo de 2005

[36] Bo Leuf and Ward Cunningham. *The Wiki Way*. Addison-Wesley Professional. ISBN: 020171499X

[37] Carnegie Mellon University - Software Engineering. Institute: *The Capability Maturity Model. Guidelines for Improving the Software Process*. SEI Series in Software Engineering. ISBN:0-201-54664-7

[38] Terry Winograd: *A Language / Action Perspectiva on the Design of Cooperative Work*. Publicado en: Human – Computer Interaction 3:1 (1987-1988), 3-30.

[39] Yahoo! Messenger. *http://messenger.yahoo.com/* Último acceso: 19 de Mayo de 2005

[40] Yahoo! Groups. *http://groups.yahoo.com/* Último acceso: 19 de Mayo de 2005