*Alla mia famiglia.*

# Indice

# Capitolo 1: Introduzione

Scopo di questa tesi è proporre un modello per l'integrazione e l'interoperabilità delle tecnologie collaborative disponibili in questo momento sul mercato. Le tecnologie considerate sono quelle che consentono a persone fisicamente distanti tra loro di collaborare mediante l'uso del computer.

Poiché oggi sono presenti un gran numero di software che supportano la collaborazione, ci si è chiesto in cosa questi software differiscano e quali caratteristiche abbiano invece in comune. L'analisi vuole dimostrare che poiché molte tecnologie forniscono lo stesso tipo di funzionalità utilizzando, però, protocolli differenti, è possibile identificare esattamente le specialità che caratterizzano ogni tecnologia e individuare quali applicazioni possono interagire tra loro. Inoltre si è notata la necessità di specificare in che modo quest'interazione venga effettuata e come le parti possano specificare le proprie caratteristiche.

A questo proposito la tesi propone un modello di collaborazione che permette di identificare univocamente le conversazioni che si possono creare utilizzando un determinato tipo di applicazione e fornisce anche un modo per specificare alcune preferenze dell'utente.

A prescindere, infatti, da quello che i diversi software permettono di fare, bisogna tenere fortemente in considerazione il fatto che utenti differenti hanno desideri e necessità diverse. Questa considerazione ci ha spinto a creare un modello in cui l'utente ha la possibilità di esprimere le sue preferenze.

Anche se è possibile ritrovare nella letteratura scientifica molti articoli che trattano la negoziazione del contenuto, non sono state trovate proposte che si occupano di rendere interoperabili tecnologie diverse. Infatti, nessuno prima d'ora ha proposto un modello che permetta di selezionare il tipo di conversazione possibile secondo il software a disposizione.
L'innovazione di questo lavoro sta nel considerare le tecnologie collaborative come un'insieme di applicazioni che possono essere trasformate da una all'altra semplicemente modificando qualche parametro.

Poiché si voleva poter distinguere questo tipo di tecnologie dalle altre disponibili, è stato definito dagli studiosi del settore un campo di studio, il Computer Supported Collaborative Work (CSCW), che dal 1984 si occupa di analizzare gli effetti sociali, psicologici e organizzativi dell'uso di applicazioni collaborative e di studiare come le persone lavorano insieme e come l'utilizzo dei groupware influisce sul comportamento del gruppo. Il termine Groupware si riferisce alle tecnologie che le persone usano per collaborare e all'insieme di hardware e software che supporta il lavoro di gruppo.
Le applicazioni per la collaborazione sono solitamente suddivise in tre sottogruppi in relazione agli scopi: comunicazione, collaborazione e cooperazione.
In questo lavoro si propone una suddivisione meno specifica che considera la collaborazione come l'intero spazio in cui le persone si muovono e la comunicazione e la cooperazione come le due parti in cui è diviso lo spazio della collaborazione. Infatti, si ritiene che la comunicazione e la cooperazione siano due forme di collaborazione che si differenziano per gli scopi e per i mezzi utilizzati.
Le tecnologie per la comunicazione sono quelle utilizzate principalmente per scambiare informazioni e che spesso permettono ad utenti molto lontani fisicamente di comunicare simulando conversazioni faccia-a-faccia.
Appartengono a questo sottogruppo molte tecnologie tra le quali: *E-mail, Forum e Newsgroup, Blog, Chat, Instant Messaging, VoIP, e Videoconferenza.*
Altre tecnologie si propongono come valido supporto alla cooperazione, ossia a tutta quella serie di attività che necessitano la partecipazione di vari utenti, ognuno dei quali aggiunge contributi utili a tutto il gruppo in accordo con gli obiettivi prefissati.

Appartengono a questa categoria *l'Editing condiviso, il Disegno condiviso, le Annotazioni, le Whiteboard condivise, i WikiWikiWeb, i Calendari, i MMORPG, i Giochi online, il Desktop Sharing e i Workflow system*.

Nel seguito ci riferiamo alla collaborazione come una o più conversazioni tra diversi partecipanti. Ogni conversazione è composta da un certo numero di espressioni, ognuna delle quali proviene da uno dei collaboratori.
Una delle assunzioni fatte per questo lavoro è che esistono un numero finito di caratteristiche che definiscono e descrivono la conversazione e quindi il tipo di collaborazione che può essere creata. In particolare, crediamo che ogni possibile conversazione possa essere completamente descritta collocandola su cinque differenti assi:

1. La relazione con la risorsa
2. Il meccanismo di turno
3. Il formato dati di ogni espressione
4. La visibilità
5. Il protocollo di update

La prima dimensione per valutare le applicazioni collaborative riguarda il rapporto tra i collaboratori e la risorsa su cui essi collaborano. Nel modello distinguiamo la collaborazione circa una risorsa, la collaborazione su una risorsa e la collaborazione che è la risorsa. Questa distinzione ci permette di descrivere un certo numero di situazioni differenti in cui la risorsa influenza lo stile della collaborazione.
Nella *collaborazione circa una risorsa*, esiste una risorsa principale ma questa non cambia durante la collaborazione poiché le espressioni inviate vi fanno solo riferimento. E' ad esempio il caso di commenti esterni ad una risorsa che permettono ai collaboratori di esprime informazioni secondarie senza modifiche al documento originale.
Quando *si collabora su una risorsa* questa assume stati diversi man mano che i collaboratori inviano le proprie espressioni. Ad esempio, gli editor collaborativi o il client MMORPG permettono di esprimere le espressioni come comandi sulla risorsa condivisa. Le applicazioni devono preservare la sequenza delle espressioni per

mantenere la consistenza del documento e devono fornire un meccanismo di lock per assicurare una corretta evoluzione del contenuto.
Quando la *collaborazione è la risorsa* non pre-esiste una risorsa vera e propria a cui associare le espressioni. Le espressioni *sono*, infatti, la risorsa su cui si collabora. Le chat, i forum o le applicazioni di gioco, tutte condividono l'idea che la conversazione e il risultato della collaborazione sono, in effetti, la stessa cosa. Se si desidera ogni sessione può essere salvata e trasformata in una risorsa vera e propria. Anche in questo caso alcune applicazioni possono dover assicurare un rigoroso meccanismo di turni, altre possono permettere un approccio più flessibile.

Il Computer Supported Collaborative Work colloca le tecnologie collaborative all'interno di una matrice bidimensionale di spazio e tempo e permette di distinguere applicazioni sincrone e asincrone e applicazioni locali e remote.
Così, ad esempio, un editor collaborativo può essere sia sincrono che asincrono secondo il tipo di applicazione in uso mentre un sistema di chat sarà necessariamente sincrono.
La distinzione tra applicazioni locali e remote, almeno nel contesto del Web, non influenza i protocolli di collaborazione, dunque questa distinzione non verrà considerata nel nostro lavoro. D'altra parte la distinzione tra applicazioni sincrone e asincrone ha una grande importanza per l'architettura tecnica necessaria all'implementazione di ogni tipo di applicazione.
Quello che proponiamo in questa tesi è un differente approccio al concetto di sincrono contro asincrono: noi consideriamo il *turno* come il set di regole per guadagnare o perdere il diritto di produrre una o più espressioni durante la conversazione.
Il meccanismo di turni è caratterizzato da due parametri: l'*esclusività* e la *numerosità* che indicano, rispettivamente, il numero di utenti che possono simultaneamente produrre espressioni e, nel caso di un lock, il numero di espressioni (una o più) che possono essere inviate durante un turno.
Alcuni esempi possono chiarire lo schema. Una applicazione sincrona di gioco di carte potrebbe garantire ad ogni giocatore un lock esclusivo per effettuare la propria mossa. Nessun altro partecipante può giocare una carta ma appena la mossa è stata eseguita il lock viene automaticamente rilasciato e passato al giocatore successivo.

Un'applicazione sincrona di editing condiviso, invece, potrebbe dare al gruppo dei partecipanti un lock locale con invio di espressioni multiple. Quando un utente sta modificando una parte può inviare quante espressioni desidera e nessun altro può modificare la stessa parte.
Ancora, una applicazione asincrona di editing potrebbe fornire un lock globale sul documento in modo che solo un partecipante alla volta possa effettuare modifiche. Appena il partecipante con il turno ha finito rilascia manualmente il lock in modo che qualcun altro possa prenderlo.
Per quanto riguarda il formato dati, il tipo di applicazione collaborativa utilizzata determina l'esatta natura dei dati scambiati. Così, una chat permette lo scambio di espressioni testuali mentre i giochi richiedono mosse legali.
La visibilità delle espressioni è un'altra dimensione importante perché ci permette di distinguere le conversazioni private dalle conversazioni pubbliche. Il primo caso è quello delle applicazioni che permettono ai partecipanti di essere sempre informati su chi sta prendendo parte alla collaborazione e può quindi ottenere le espressioni inviate dagli altri partecipanti. Sono un esempio le chat, la videoconferenza o l'instant messaging.
Nel secondo caso, invece, la visibilità pubblica ci permette di esprimere che non è possibile identificare tutti i partecipanti. Ad esempio nei forum, nei blog o nei wiki non è possibile sapere chi sono i destinatari di un'espressione.
Per quanto riguarda il meccanismo di aggiornamento abbiamo determinato tre modalità; esse sono: *pull* quando l'utente richiede manualmente gli aggiornamenti; *pseudopush* quando è il sistema a fare richiesta delle nuove espressioni ad ogni tot di tempo fissato dall'utente; *realpush* quando le espressioni vengono inviate non appena disponibili senza bisogno di richiederle. Per approfondire questa suddivisione si rimanda al capitolo 4 dove le dimensioni sono analizzate nei particolari e dove è possibile trovare un'analisi dettagliata delle tecnologie collaborative relativamente alle dimensioni proposte.

Nella creazione del nostro modello che chiamiamo **CSI** (Collaborative Services Initiative) il secondo passo è stato quello di analizzare le soluzioni attualmente disponibili per la negoziazione. Il processo secondo cui le parti possono accordarsi sulle

proprie capacità collaborative si avvicina molto al concetto di negoziazione del contenuto proposto nel contesto del "Device Independence" di cui si parlerà meglio nel Capitolo 3.
La negoziazione è un argomento molto importante di cui tener conto quando, come nel nostro caso, si parla di entità diverse che vogliono scambiare informazioni (comunicare o cooperare) e mettersi d'accordo in base alle proprie capacità.
Nel contesto della negoziazione le parti che vogliono collaborare spesso hanno a disposizione hardware e software di diversi produttori e per questo i documenti di una parte potrebbero non essere leggibili o visualizzabili dall'altra parte.
Per questo motivo le parti hanno necessità di un'architettura software che permetta loro di specificare cosa siano in grado di fare ed eventualmente di proporre le proprie preferenze.
Nella tesi analizzeremo il funzionamento della negoziazione nel protocollo HTTP, poi ci sposteremo su una sua variazione, il Transparent Content Negotiation (TCN), proseguiremo con la descrizione del protocollo CC/PP (Composite Capability/Preference Profile), ed infine analizzeremo ebXML. I primi tre approcci analizzati sono protocolli e tecnologie nate nel contesto del Device Independence perciò riguardano in particolar modo il problema di negoziare le caratteristiche delle parti per poter adattare il contenuto web non solo per quanto riguarda l'architettura hardware e software ma anche per quanto riguarda le preferenze dell'utente. ebXML invece è uno standard per l'e-commerce che si propone di fornire modalità d'interazione tra "business information system" .

Il nostro lavoro è stato quello di creare dei protocolli finalizzati alla standardizzazione e all'integrazione del gran numero di applicazioni collaborative correntemente disponibili sul Web.
Abbiamo stabilito la necessità di almeno due protocolli: il CAP (Collaboration Agreement Protocol) che permette ai partecipanti di accordarsi sul tipo di collaborazione da effettuare in base alle dimensioni citate precedentemente e il CUML (Collaborative Utterance Markup Language) che permette ai partecipanti di contribuire con delle espressioni in accordo con il profilo accettato da tutti i collaboratori e distribuito possibilmente attraverso un server collaborativo.

Il protocollo CAP permette alle parti di stabilire un framework collaborativo in cui le regole sono descritte esplicitamente.
Ogni collaboratore che propone di creare una nuova conversazione o che viene invitato a parteciparvi deve proporre le proprie caratteristiche e le proprie preferenze. Il server a questo punto deve occuparsi del confronto tra i profili di tutti i collaboratori e restituire il più ampio set di caratteristiche comuni. Ogni partecipante in seguito può decidere se partecipare alla conversazione o disconnettersi completamente.
Il numero di conversazioni all'interno della stessa sessione collaborativa può essere vario. Ad esempio una raffinata applicazione di editing condiviso potrebbe mantenere una conversazione di comandi sul testo più una conversazione di chat tra gli autori più un'ulteriore conversazione di commenti su parti specifiche del documento. Ogni conversazione è totalmente indipendente dalle altre e può avere caratteristiche completamente differenti.
Il protocollo CUML si occupa, invece, del formato dei messaggi con contenuto che i collaboratori si scambiano dopo essersi messi d'accordo.
Ogni contributo individuale alla conversazione è chiamato espressione. Ogni espressione deve fornire informazioni di contesto ogni volta che viene ricevuta dal server collaborativo. Esempi di queste informazioni sono: a quale conversazione l'espressione appartiene, chi la fornisce e quali certificati usa o la risorsa (se c'è) a cui l'espressione si riferisce.
Il protocollo CUML fornisce alle applicazioni il modo in cui esprimere questo tipo di informazioni. Esso cattura la natura di ogni espressione e il tipo di dato (che come detto può andare dal testo semplice nelle sessioni di chat agli oggetti strutturati, come gli elementi SVG, in ambienti di disegno condiviso) di ogni espressione.
Il CUML, inoltre, non limita la granularità dei dati che vengono scambiati che possono andare dal singolo carattere in una applicazione per l'editing testuale sincrono ad una intera frase in una chat o in un forum. Come detto, questi parametri vengono decisi nel CAP.
Un'altra caratteristica fondamentale e innovativa di questa proposta è la possibilità di rinegoziare dinamicamente l'accordo.
La possibilità di modificare in ogni momento i parametri di collaborazione permette di aumentare l'utilità, l'usabilità e la flessibilità delle applicazioni.

Ad esempio, i partecipanti ad una conversazione sincrona di chat potrebbero aggiungere una cartolina di disegno comune (creando una nuova conversazione), potrebbero salvare la conversazione (trasformando la chat in un forum) o potrebbero invitare qualcun altro a partecipare.
Allo stesso modo un editor collaborativo sincrono potrebbe essere trasformato in asincrono semplicemente modificando il lock da locale a globale.
La rinegoziazione dinamica dell'accordo è esattamente la caratteristica che permette di associare tra loro applicazioni differenti e che rende possibile la trasformazione di un'applicazione nell'altra semplicemente modificando qualche regola di collaborazione.

# Capitolo 2: CSCW e Tecnologie per la collaborazione

In questo capitolo sarà descritta la nascita del termine CSCW e la sua storia e saranno analizzate individualmente le tecnologie Groupware disponibili sul mercato.

## 2.1 CSCW: Computer Supported Collaborative Work

Il modo in cui gli uomini possono lavorare in cooperazione non è un campo di studio recente ed è tuttora molto attivo. In particolare, negli anni Ottanta, una serie di fattori tecnologici e sociali fece crescere l'interesse per i sistemi informatici a supporto del lavoro cooperativo. Da una parte ci fu la diffusione dei personal computer negli uffici e la possibilità di connettersi in rete; dall'altra un mercato che diventava molto competitivo ebbe bisogno di nuove forme d'organizzazione del lavoro orientate verso le attività di gruppo. Questi fattori diedero una spinta notevole alla nascita di software a supporto della collaborazione.

Il termine CSCW fu coniato dalla dottoressa Irene Greif del MIT e da Paul Cashman della Digital che nel 1984 organizzarono una conferenza sullo sviluppo di sistemi che avrebbero aiutato le persone nel loro lavoro attraverso l'uso del computer.

In quest'ambiente si formò rapidamente una comunità scientifica molto numerosa che diede vita a tre convegni internazionali nel giro di pochi anni; il primo convegno si svolse nel 1986 in Texas, poi altri due nel 1988 e nel 1989.

In questi anni l'interesse per questo nuovo campo della scienza è cresciuto notevolmente sia da parte del mondo dell'informatica sia da parte di psicologi, sociologi e antropologi. Ogni anno, ad anni alterni in Europa e Nord America, si svolge una conferenza sui sistemi collaborativi e CSCW.

Ancora oggi a causa dell'interesse delle più disparate categorie di ricercatori è difficile spiegare precisamente cosa siano in realtà Groupware e CSCW.
Alcune definizioni, tra le più popolari, sono le seguenti:

> "*Groupware are computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.*" Ellis et al. [EGR91]

> "*CSCW is the study of how people work together using computer technology. Typical topics include use of email, hypertext that includes awareness of the activities of other users, videoconferencing, chat systems, and real-time shared applications, such as collaborative writing or drawing*." Brinck, Gomez [BriGom92]

In accordo con queste definizioni il termine Groupware si riferisce alle tecnologie che le persone usano per collaborare e all'insieme di hardware e software che supporta il lavoro di gruppo; CSCW studia gli effetti sociali, psicologici e organizzativi dell'uso di queste tecnologie e analizza come le persone lavorano insieme e come l'utilizzo dei groupware influisce sul comportamento del gruppo.
I groupware sono solitamente collocati in una matrice spazio-temporale[EGR91][Gri00][Ban93].
Come mostrato in figura 1, questa classificazione permette di distinguere, da una parte, collaborazioni sincrone e asincrone e dall'altra, collaborazioni locali e remote.
Una collaborazione sincrona avviene in tempo reale, come durante una chat o una videoconferenza. I partecipanti hanno accesso alle stesse informazioni nello stesso momento.

In una collaborazione asincrona lo scambio avviene in tempi più lunghi, come nell'uso di email o nell'aggiunta di messaggi su un forum.
I groupware che supportano la collaborazione sincrona sono detti *real-time*, altrimenti *non-real-time*.

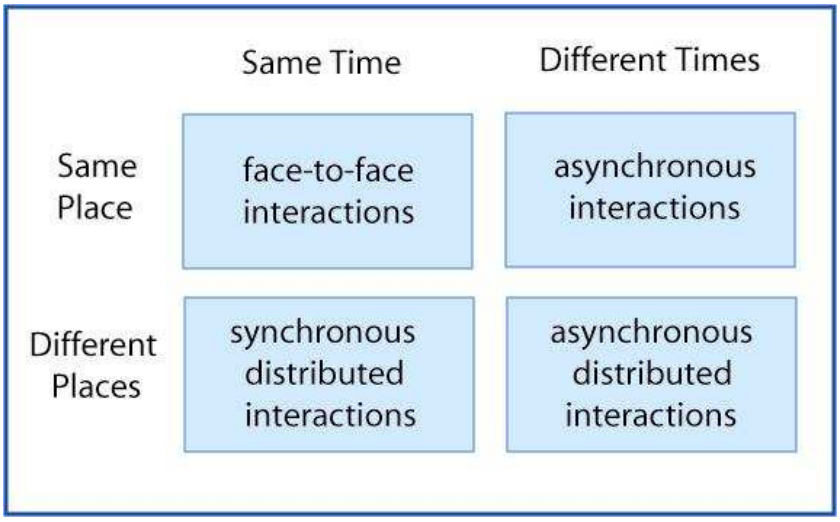


**Figura 1**: Matrice spazio-temporale

Durante il loro sviluppo, i groupware si sono evoluti da strumento per la comunicazione tra singoli (attraverso le e-mail) a supporto per il lavoro di gruppo fino a diventare una necessità per la coordinazione e la comunicazione anche all'interno di grandi aziende.
Attualmente, grazie a queste tecnologie, milioni di persone possono scambiare opinioni, dati e conoscenze o semplicemente rimanere in contatto con conoscenti lontani.
Le tecnologie groupware sono solitamente suddivise in tre sottogruppi: comunicazione, collaborazione e cooperazione[EGR91].
Molta confusione nel campo del CSCW dipende dalle diverse interpretazioni dei termini collaborazione e cooperazione. Alcuni autori[DBBO96] fanno una netta distinzione:

> *"Cooperation and collaboration do not differ in terms of whether or not the task is distributed, but by virtue of the way in which it is divided; in cooperation the task is split (hierarchically) into independent subtasks; in collaboration cognitive processes may be (heterarchically) divided into intertwined layers. In cooperation, coordination is only required when assembling partial results,*

*while collaboration is a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem."*

Noi proponiamo una suddivisione meno specifica che considera la collaborazione come l'intero spazio in cui ci si muove e la comunicazione e la cooperazione come le due parti in cui lo spazio è diviso. Questo perché si ritiene che la comunicazione e la cooperazione siano due forme di collaborazione che si differenziano per gli scopi e per i mezzi utilizzati. In generale, comunque, sarà mantenuta questa suddivisione solo nella descrizione seguente perché dal lato pratico non vi sono sostanziali differenze tra tecnologie per la comunicazione o per la cooperazione.
Nel prossimo paragrafo saranno descritte le principali tecnologie disponibili in questo momento e saranno analizzate le loro caratteristiche.

### 2.1.1 Comunicazione

Le tecnologie per la comunicazione sono quelle utilizzate principalmente per scambiare informazioni e che spesso permettono ad utenti molto lontani fisicamente di comunicare simulando conversazioni faccia-a-faccia.
Appartengono a questo sottogruppo le seguenti tecnologie: *E-mail, Forum e Newsgroup, Blog, Chat, Instant Messaging, VoIP, e Videoconferenza*[Gri00].

#### *2.1.1.1 Email*

La posta elettronica è il più chiaro esempio di una tecnologia groupware che ha avuto un grande impatto sia dal punto di vista tecnologico che da quello sociologico. La possibilità di scambiare messaggi con altre persone attraverso la rete ha dato senza dubbio uno slancio alla creazione di gruppi di collaborazione composti da persone fisicamente distanti tra loro.
Inizialmente nata per il semplice scambio di messaggi tra due persone, la posta elettronica oggi include diverse funzioni quali la notifica, l'inclusione di allegati, la criptazione e la firma digitale, l'utilizzo di agenti per l'automazione della gestione dei

messaggi; inoltre semplicemente inserendo più indirizzi nel campo Destinatario è possibile inviare lo stesso messaggio ad un numero variabile di persone.
I seguenti protocolli sono fra i più comunemente utilizzati per il trasferimento di e-mail da un sistema all'altro.
La consegna della posta da una applicazione client al server, e da un server originatore al server destinatario, è gestita dal *Simple Mail Transfer Protocol* (*SMTP*)[Kle01].
Il protocollo SMTP non richiede autenticazione. Questo permette a chiunque su Internet di inviare email a chiunque, anche ad un gruppo molto grande di persone. Questa caratteristica di SMTP è la causa principale della posta indesiderata o *spam*.
Il *Post Office Protocol*(*POP3*)[MyeRos96] e l'*Internet Message Access Protocol* (*IMAP*)[Cri04] sono utilizzati dalle applicazioni client per scaricare le email dai server di posta. Diversamente da SMTP entrambi questi protocolli richiedono un nome utente e una password per poter accedere al server e per entrambi i protocolli le password passano in chiaro attraverso la rete.

#### *2.1.1.2 Forum e Newsgroup*

I newsgroup sono lo strumento di discussione originario di Arpanet[1], la rete delle università USA da cui deriva Internet.
I newsgroup sono gestiti dai server di news cioè da quella rete di calcolatori che forma la cosiddetta area Usenet[2] di Internet. I Newsgroup non fanno riferimento ad un sito particolare ma ci si possono appoggiare e hanno una visibilità molto ampia, specialmente se in lingua inglese. Per accedervi in modo ottimale spesso vengono utilizzati programmi specializzati (*newsreader*) che invece di appoggiarsi ad un sito web sono in grado di collegarsi direttamente alla rete Usenet.
Alcuni newsreader richiedono di restare collegati al server news durante la lettura, altri invece consentono di scaricare velocemente i messaggi sul disco per leggerli anche

[1] Arpanet: (Advanced Research Projects Agency Network) venne studiata e realizzata nel 1969 dal DARPA (Defence Avanced Research Project Agency) del ministero della difesa degli Stati Uniti. Per tutti gli anni Settanta, ARPAnet si sviluppò in ambito universitario e governativo, e dal 1974 con l'avvento dello standard di trasmissione TCP/IP, il progetto iniziò ad essere denominato Internet. Con il passare del tempo, l'esercito si disinteressò sempre più al progetto che rimase sotto il pieno controllo delle università, diventando un utile strumento per scambiare le conoscenze scientifiche e per comunicare.

[2] Usenet: rete mondiale formata da migliaia di newsgroup.

offline. Il Network News Transport Protocol, spesso abbreviato come NNTP[KanLap86], è il protocollo usato dai newsgroup.
I forum, chiamati anche gruppi di discussione, sono integrati direttamente all'interno di un sito Web e permettono ai partecipanti di aggiungere messaggi in tema con l'argomento del forum e del thread nel quale è inserito.
I forum forniscono immediatamente il contesto di ogni discussione favorendo in questo modo l'inserimento dei nuovi arrivati che possono rapidamente rendersi conto sia dell'argomento oggetto del dibattito sia delle opinioni espresse in proposito. In genere, i software o i servizi online per la gestione dei forum consentono di archiviare tutti i messaggi inseriti, altri software invece cancellano i messaggi anteriori ad una certa data. Conservare i testi in archivio è, di fatto, una delle funzioni più utili, perché si può avere costantemente presente tutta la storia della comunità e la sua evoluzione nel tempo.
Più recente è la possibilità di inserire immagini o attivare link direttamente all'interno del testo.

#### *2.1.1.3 Blog*

I Blog (contrazione di *web log*) sono siti web in cui l'autore aggiunge periodicamente dei messaggi (*post*) che possono essere letti da chiunque.
I post, tipicamente visualizzati in ordine cronologico inverso, sono principalmente testuali e talvolta possono contenere foto o altri oggetti multimediali.
Herring, Scheidt, Bonus e Wright[HSBW04] hanno individuato tre tipi di blog: personali, che tipicamente riguardano la vita dell'autore (si può quindi vedere come un diario moderno); i filtri, blog in cui l'autore seleziona contenuto da altri siti web e aggiunge le sue opinioni; e i blog informativi in cui lo scopo principale è mettere in luce argomenti d'interesse comune che a parere dell'autore non vengono sufficientemente trattati dai comuni media informativi.
In generale, in tutti i tipi di blog il visitatore può interagire commentando i messaggi dell'autore e aggiungendo le sue opinioni.

#### *2.1.1.4 Chat*

Le Chat (abbreviazione di *Conversational Hypertext Access Technology)* permettono la comunicazione interattiva real-time tra varie persone all'interno di *chat rooms.* Quando in una chat un utente scrive un messaggio questo è istantaneamente disponibile a tutti gli altri partecipanti o, se si è in una stanza privata, solo ad alcuni. Oltre alla visualizzazione dei messaggi sono solitamente disponibili alcune altre funzionalità come la possibilità di inviare messaggi ad un unico partecipante o cambiare room. Inoltre la maggior parte delle chat fornisce nella finestra di visualizzazione la lista degli utenti presenti all'interno della stessa room.

Internet Relay Chat (IRC)[OikRee93] è uno dei più vecchi sistemi di chat ancora in uso e consiste in una rete di server a cui i client si connettono usando la porta 6667.
IRC è un protocollo di rete aperto che utilizza il protocollo di trasmissione TCP (Transmission Control Protocol) e opzionalmente l'SSL (Secure Sockets Layer). Un server IRC (chiamato IRCd) è in grado di connettersi con altri server IRC formando così una vera e propria rete di comunicazione. Molti server IRC non richiedono un identificativo utente, ma prima di collegarsi l'utente deve comunque impostare un nickname. Anche se molto utilizzata, la rete IRC soffre di alcuni problemi, come la complessità d'uso, che hanno portato alla crescita nell'utilizzo delle *Web Chat.*
Le web chat sono molto simili ai client di chat ma non è necessaria l'installazione di alcun software dedicato, basta infatti accedere ad uno dei tanti siti che offrono questo tipo di servizio. Le Web Chat sono molto semplici da usare, l'interfaccia è familiare anche ai nuovi utenti che non hanno la necessità di imparare ad utilizzarla.
Sono presenti in rete molti tipi di Web Chat, ognuna basata su un protocollo diverso. Dewes, Wichmann e Feldmann[DWF03] distinguono tre tipi di questi sistemi:

1. *HTML-Web-Chat*: usano il browser come interfaccia utente e HTTP come protocollo a livello applicazione;
2. *Applet-Web-Chat*: l'interfaccia utente è un'applet;

3. *Applet-IRC-Chat*: sistemi che usano un'applet Java o Javascript come interfaccia ad IRC. Il client converte ogni messaggio che l'utente inserisce nell'applet in un messaggio IRC e viceversa.

Per assicurare che ogni utente ottenga l'output corretto molti sistemi usano degli ID di sessione, cioè una stringa o un numero, in qualche modo derivanti dal processo di login, che viene aggiunto in ogni richiesta o risposta, a volte utilizzando un cookie[3].

#### *2.1.1.5 Instant Messaging*

Nata nel 2001, questa recente tecnologia è cresciuta rapidamente. Utilizzata inizialmente dai giovani per scopi sociali[GriPal02][IWWSK02], oggi è utilizzata anche nelle aziende, raccogliendo la stima di molti milioni di utenti.
L'instant messaging permette di comunicare rapidamente in real-time ma a differenza della chat ogni utente ha una lista di contatti (*Buddy list*) ed è possibile controllare la disponibilità dei propri amici o colleghi (ad esempio *online, away, offline*), essere avvisati appena uno dei contatti si connette, e comunicare immediatamente. Alcune applicazioni di instant messaging danno inoltre la possibilità di usufruire di un servizio analogo agli SMS: se un utente contattato non è connesso, il server memorizza il messaggio e lo recapita all'utente chiamato appena si connette.
I moderni sistemi di instant messaging consistono in un server a cui si collegano i programmi client: il server tiene traccia di quali computer e quali utenti sono connessi al sistema e gestisce le comunicazioni fra i vari client. Per motivi di sicurezza non viene fornito, in generale, l'indirizzo di rete delle parti in comunicazione, tuttavia, per migliorare efficienza durante lo scambio di file oppure di flussi audio e video, il server comunica a ciascun client l'indirizzo di rete per permettere ai due di scambiarsi i dati direttamente senza passare per il server.
Esistono diversi protocolli per l'instant messaging, ad esempio: SIMPLE[CRSHG02] (Session Initiation Protocol for Instant Messaging and Presence Leveraging

[3] I cookie (letteralmente "biscottini") sono piccoli file di testo inviati al computer di un visitatore di un sito web che registrano informazioni riguardanti le attività svolte durante la navigazione. Alcuni siti web utilizzano i cookie per identificare i visitatori e per visualizzare informazioni più personalizzate a una successiva connessione. Gli utenti possono accettare o rifiutare l'invio di cookie modificando le opzioni del proprio browser.

Extensions), APEX[RKC02] (Application Exchange), OSCAR[Kuh02](Open System for CommunicAtion in Realtime) e XMPP[Sai04], comunemente conosciuto come Jabber[Jab]; tuttavia anche se aziende, utenti e provider sono d'accordo sul fatto che sarebbe una buona idea adottare un unico protocollo standard per tutti i sistemi di messaging, nessuna azienda ha un vero interesse a rendere compatibile la propria rete con le altre.

Esistono alcuni client di instant messaging, quasi sempre open source[4], che sono in grado di usare contemporaneamente due o più protocolli proprietari permettendo di essere connessi a più reti di messaging tramite un solo programma client, ad esempio Trillian[TRI], Gaim[GAI] e Kopete[KOP]; molti altri client invece supportano esclusivamente un solo protocollo come, ad esempio, ICQ[ICQ], AOL Instant Messenger[AIM], o Yahoo! Messenger[YIM].

La conversazione nativa degli instant messaging è uno-a-uno, ma viene data la possibilità di estenderla attraverso gli inviti. Durante una normale conversazione tra due persone, altre possono unirsi, e creare una sorta di chat, semplicemente rispondendo ad un invito da parte di uno dei due utenti iniziali.

Secondo il programma in uso, le funzionalità andranno dal solo invio simultaneo di messaggi testuali, alla possibilità di inviare file, o creare conversazioni audio e video.

#### *2.1.1.6 VoIP*

VoIP (*Voice over Internet Protocol*) è una tecnologia che permette di effettuare una conversazione vocale sfruttando una connessione internet anziché passare attraverso la normale linea di trasmissione telefonica.

Le reti IP non dispongono di alcun meccanismo in grado di garantire che i pacchetti di dati saranno ricevuti nello stesso ordine in cui sono stati trasmessi, né alcuna garanzia relativa, in generale, alla qualità del servizio. Le attuali applicazioni VoIP si trovano a dover affrontare problemi di latenza (sostanzialmente si deve ridurre il tempo di transito e d'elaborazione dei dati durante le conversazioni) e di integrità (prevenire perdite e danneggiamenti delle informazioni contenute nei pacchetti).

[4] Software rilasciato insieme ai sorgenti in modo che qualsiasi programmatore possa apportare modifiche a condizione di ridistribuirlo con il codice sorgente. E', inoltre, la filosofia che ha permesso a migliaia di programmatori sparsi per il mondo di creare il sistema operativo GNU/Linux.

Oggi sono presenti due modelli[Mat05] per le applicazioni VoIP; da una parte il modello con server centralizzato chiamato peer-to-peer ibrido, adottato dall'IETF (Internet Engineering Task Force) con il suo protocollo SIP, *Session Initiation Protocol* [RSCJPSH02], dall'altra il modello peer-to-peer puro, di cui Skype[Skype] è il maggiore esponente.
La caratteristica dell'approccio ibrido è la presenza di un server a cui gli utenti si devono connettere quando hanno bisogno di un servizio come una chiamata tramite internet o una videoconferenza. SIP è un protocollo di livello applicazione che può stabilire, modificare e terminare sessioni multimediali tra due o più partecipanti. Le sessioni possono essere conferenze, messaggi istantanei o chiamate telefoniche attraverso la rete.
Poiché il ruolo del server è quello di inoltrare le richieste dei client ha la necessità di conoscere l'indirizzo della destinazione. Per ottenere queste informazioni sono necessari più server; in particolare l'architettura SIP è composta, oltre che dallo User Agent (tipicamente un telefono o un gateway), da tre server per ogni dominio: Proxy Server, Registrar Server e Redirection Server. Il server di registrazione è importante perché permette che altri utenti possano trovare l'indirizzo IP del destinatario e comunicare il proprio. I server Proxy sono utili per trasmettere le richieste dei client verso i destinatari mentre i server di Redirezione comunicano agli User Agent informazioni su come raggiungere il destinatario.

Anche l'approccio peer-to-peer puro adottato da Skype permette chiamate vocali, scambio di messaggi testuali e conferenza. L'architettura Skype è composta da tre entità[Mat05]:

- *Nodi host*: Applicazione Skype, che può essere usata per chiamate telefoniche e invio di messaggi;
- *Super nodi*: Ogni nodo della rete con un IP pubblico e le necessarie risorse in termini di CPU, memoria e capacità di banda può essere un super nodo. I supernodi sono quindi dei nodi skype ordinari ma non devono essere protetti da un firewall. Nel sito di Skype si legge:

> "*A Supernode is a computer running Skype Software that has been automatically elevated to act as a hub. Supernodes may assist in helping other users to communicate or use the Skype Software efficiently. This may include the ability for your computer to help anonymously and securely facilitate communications between other users of the Skype software who, due to network and firewall constraints, cannot establish direct connections.*"

- *Server di login*: Gli user name e le password sono salvate su questo server; ogni utente deve registrarsi e connettersi al server di login per essere autenticato quando si connette alla rete Skype.

A differenza del login, stabilire una chiamata, trasferire dati e chiudere la chiamata sono funzioni decentralizzate. Ogni nodo contiene una lista di super nodi aggiornata frequentemente e deve essere presente almeno una entry (indirizzo IP e porta di un nodo Skype online) affinché ci si possa connettere alla rete Skype.
Quando un utente vuole effettuare una chiamata, l'applicazione client deve connettersi inizialmente ad un super nodo e in seguito deve registrarsi al server di login con uno user name e una password.
Skype utilizza dei codec che permettono di mantenere una qualità soddisfacente delle chiamate e usa sia TCP che UDP per gestire il traffico. In particolare il client rimane in ascolto su una porta TCP e una porta UDP secondo quanto stabilito nella configurazione e apre le porte 80 (HTTP) e 443 (HTTPS). Skype inoltre è in grado di attraversare NAT e firewall [BasShu04].

#### *2.1.1.7 Videoconferenza*

La videoconferenza permette a due o più partecipanti di effettuare una conversazione audio e video in tempo reale. Si definisce videoconferenza la combinazione di due tecnologie: la videotelefonia, che permette di vedere soltanto il proprio interlocutore e la videoconferenza vera e propria, che permette a più persone di condividere un unico canale di comunicazione.

Oltre alla possibilità di vedere il proprio interlocutore, la videoconferenza permette a volte di disporre di un pannello di controllo dove sono indicati i partecipanti e uno spazio di lavoro virtuale comune, in cui tutti i partecipanti possono condividere testi, immagini, tabelle ed altre informazioni. La condivisione di grafici o programmi può essere supportata mediante applicazioni di whiteboarding che sono spesso incluse nei pacchetti di videoconferenza.

La videoconferenza è stata a lungo una chimera, presentata alle esposizioni universali di Bruxelles nel 1958 e di Montreal nel 1967, ed oggetto di alcune costose e poco efficienti installazioni pilota. Il motivo di questa lenta evoluzione è da ricercare nella limitata capacità di trasporto delle linee di rame tradizionali, limitazione che è stata progressivamente superata a partire dal 1995 man mano che cominciavano a diffondersi le linee a fibra ottica.

Ottenere adeguate prestazioni tecniche era solo uno dei problemi; l'altro non meno importante era individuare possibili e utili applicazioni nel mondo reale.

La videoconferenza, oggi, consente alle piccole e grandi imprese l'organizzazione di incontri virtuali senza costi di viaggio e con un preavviso molto breve. Le applicazioni in questo ambito consentono meeting estremamente realistici e l'elemento visivo facilita la discussione su oggetti fisici come prodotti o diagrammi.

Esistono vari standard e protocolli a cui si attengono i costruttori e gli operatori per garantire l'interoperabilità dei sistemi di videoconferenza. Fra i principali protocolli, in funzione del tipo di rete utilizzata, si annoverano:

- il protocollo H324 su reti pubbliche commutate;
- il protocollo H.320. Su reti ATM;
- il protocollo H.321su reti ISDN;
- il protocollo H.323 o, in alternativa, il protocollo SIP su reti internet IP;
- il protocollo H.324M su reti UMTS.

Attualmente la videoconferenza permette lo scambio di audio, video, documenti o immagini, attraverso altre tecnologie come le webcam, la telefonia internet, le whiteboard o le chat. Come nell'instant messaging inoltre è possibile in ogni momento sapere chi è collegato alla rete e disponibile.

### 2.1.2 Cooperazione

Alcune tecnologie si propongono come valido supporto alla cooperazione, ossia a tutta quella serie di attività che necessitano la partecipazione di vari utenti, ognuno dei quali aggiunge contributi utili a tutto il gruppo in accordo con gli obiettivi prefissati.
Appartengono a questa categoria *l'Editing condiviso, il Disegno condiviso, le Annotazioni, le Whiteboard condivise, i WikiWikiWeb, i Calendari, i MMORPG, i Giochi online, il Desktop Sharing e i Workflow system*.

#### *2.1.2.1 Editing condiviso*

L'editing è una classe importante di tool collaborativi, che permette a più utenti di vedere e correggere simultaneamente un documento condiviso, e di osservare le modifiche fatte dagli altri. In particolare, gli editor testuali devono supportare non solo le operazioni di base di un editor single user, ma devono anche fornire ulteriori funzionalità[Pra99]:

- *Collaboration awareness*: devono essere fornite informazioni di contesto sufficienti affinché ogni partecipante sia consapevole delle attività degli altri membri del gruppo;
- *Fault-tolerance*: si deve garantire la possibilità di ripristino della sessione in caso di problemi hardware;
- *Concurrency control*: necessario per assicurare la consistenza del documento durante modifiche simultanee;
- *Multi-user undo*: gli utenti devono poter annullare individualmente le proprie modifiche;
- *Usable as a single-user editor*: deve essere possibile utilizzare l'editor comodamente anche nel lavoro singolo.

Alcuni editor permettono di lavorare esclusivamente in modo asincrono, altre invece non possono supportare anche la collaborazione sincrona. Alcuni editor permettono non

solo la collaborazione sul testo ma forniscono anche una sorta di chat per poter scambiare con gli altri partecipanti commenti, correzioni e suggerimenti.
Solitamente il documento è diviso in segmenti di granularità arbitraria, in modo tale che mentre un utente sta modificando una parte, solo questa è bloccata, consentendo la modifica libera delle altre parti[EGR91].

Un altro tipo importante di Editor collaborativi è il sistema grafico basato sugli oggetti. Le forme come linee, rettangoli, cerchi, o contenitori di testo sono sottoposti alle modifiche concorrenti dei partecipanti e hanno delle specifiche proprietà che permettono di distinguerle e di metterle in relazione tra loro. Alcuni esempi di caratteristiche sono il tipo, la dimensione, la posizione o il colore.

#### *2.1.2.2 Annotazioni*

Schroeter, Hunter, e Kosovic[SHK04] definiscono le annotazioni come commenti, note, spiegazioni od osservazioni esterne che possono essere allegate ad un documento o ad una parte selezionata di un documento senza modificarlo.
Quando un utente richiede un documento può ottenere anche le annotazioni allegate ad esso; attraverso una semplice richiesta ad un server (*annotation server)* può consultarle, modificarle o aggiornarle.
Le annotazioni hanno grande importanza, secondo Weng e Gennari[WenGen04], nella revisione dei documenti condivisi da vari co-autori in particolare nella scrittura collaborativa asincrona.
Tra i vari software disponibili oggi sul mercato, citiamo Annotea[Ann].
Annotea è un sistema Web-based di annotazioni condivise che sono trattate come metadati. Le annotazioni sono viste come dichiarazioni dell'autore su un documento web, sono esterne al documento e possono essere salvate in uno o più annotation server.
I metadati, in generale, permettono di descrivere le proprietà delle risorse e possono essere estratti dalla risorsa stessa (ad esempio i dati contenuti nella sezione HEAD di un documento HTML), da un'altra risorsa, o possono essere trasferiti con il documento dal server al client (GET) e dal client al server (PUT o POST).

I metadati sulle annotazioni danno informazioni come la data di creazione, il nome dell'autore, il tipo dell'annotazione, l'URI del documento annotato, e un XPointer che specifica a quale parte del documento l'annotazione si riferisce.
Le caratteristiche fondamentali di Annotea, descritte da Kahan e Koivunen[KanKoi01] sono:

- *Open tecnologies*: Annotea si basa su standard aperti per semplificare l'interoperabilità con altri sistemi per le annotazioni e per massimizzare l'estensibilità dei dati che questi sistemi utilizzano;
- *Annotated documents are well-formed and structured*: Le risorse annotabili sono quelle che hanno una struttura, vale a dire documenti HTML o basati su XML;
- *Annotations are first class Web resources*: Come ogni altra risorsa sul web, le annotazioni sono associate ad un URI;
- *Annotations are typed*: Nello stesso modo in cui un'annotazione può essere vista come un metadato del documento a cui è associata, così l'annotazione stessa ha delle proprietà distinte. Ad esempio il tipo di un'annotazione è un metadato, e permette agli utenti di classificare le annotazioni mentre le creano ('*questa annotazione è un commento/ un'errata corrige'*);
- *Annotations type can be defined by users*: Permette ad ogni utente di creare i propri tipi;
- *Annotations properties must be described with an RDF schema*;
- *Annotations are stored in generic RDF database*: Importante perché permette agli utenti di riusare le informazioni salvate in un database senza doverle modificare;
- *No assumption on user interface*: Annotea descrive come i metadati possono essere associati ad un documento e come interrogare un database RDF. Non specifica come lo User Agent deve presentare i metadati all'utente;
- *Local (private) and remote (shared) annotations*: Le annotazioni possono essere salvate sia localmente sul computer dell'utente, sia su un annotation server;
- *Multiple annotations server*: Non c'è un solo server centralizzato, ma gli utenti possono facilmente configurare un server per le annotazioni e definire chi ha il permesso di consultarlo.

Ogni comunicazione tra il server e il client avviene attraverso i metodi standard di HTTP(Figura 2).

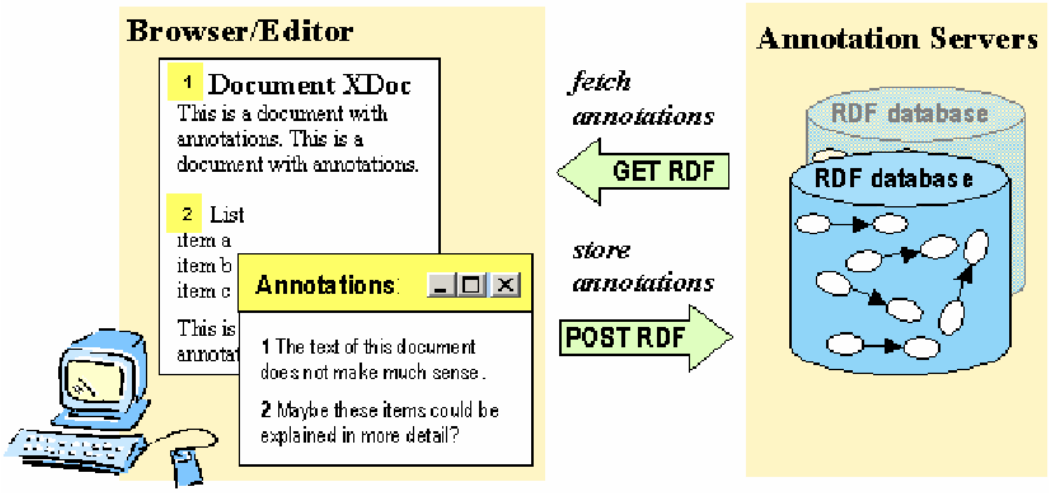


**Figura 2:** Architettura di Annotea.

### *2.1.2.3 Whiteboard condivisa*

Le shared witheboard permettono a più partecipanti in diversi luoghi di vedere simultaneamente documenti o immagini e annotarli attraverso le funzioni messe a disposizione dal software della whiteboard. I disegni possono essere discussi, corretti ed analizzati mentre vengono creati.

Ogni partecipante può sapere cosa stanno scrivendo gli altri o cosa stanno puntando con il telepuntatore[EGR91], che assume colori diversi o diverse etichette testuali a seconda dell'utente che lo muove.

Ad esempio, la suite MBONE[WB02] contiene un programma chiamato "WB" che fornisce una shared whiteboard. Il programma è progettato per utilizzare il supporto IP multicasting di MBONE ma funziona egregiamente anche nelle connessioni unicast punto-punto.

WB fornisce un set completo di tool per il disegno e una lista di partecipanti che permette di sapere in ogni momento chi sta lavorando sulla whiteboard. Inoltre ha anche funzioni di sicurezza che impediscono ai partecipanti senza password di accedere alla conversazione.

Un altro esempio di whiteboard condivisa è “ShowMe Whiteboard” che fa parte della suite di SUN Microsystem[Show99].
ShowMe Whiteboard fornisce un gran numero di tool per il disegno bitmap, mette a disposizione la cattura delle immagini a schermo e permette semplicemente trascinando la finestra di una qualunque applicazione sopra la finestra della whiteboard di far vedere agli atri partecipanti il suo contenuto.
Ancora, Collage, un prodotto della NCSA[Gra92], è un ambiente collaborativo che fornisce sia la whiteboard che l’editing testuale. Collage è ancora in versione beta e contiene ancora molti bug ma nonostante questo è una piattaforma molto stabile.
Collage offre la possibilità di catturare porzioni dello schermo per poter lavorare in finestre separate e permettere lo scambio e la condivisione di immagini animate.

#### *2.1.2.4 WikiWikiWeb*

Wiki è una parola della lingua Hawaiana che significa veloce. Ben si adatta a quello che in termini informatici è un sito web in cui i visitatori possono velocemente modificare le pagine a proprio piacimento.
Il WikiWikiWeb fu sviluppato nel 1994 da Ward Cunningham e risultava avvantaggiato rispetto ai web server collaborativi in quanto permetteva non solo di modificare il contenuto di una pagina, ma anche la sua struttura[CDP04], organizzando il contenuto.
Lamb[Lam04] evidenzia alcuni principi fondamentali dei Wiki:

- *Anyone can change anything*: Usando semplicemente un browser web, senza alcun tipo di estensione aggiuntiva, ogni utente di un sito wiki può editare qualunque pagina semplicemente ciccando sul tasto ‘EditPage’, o creare nuove pagine;
- *Wikis use simplified hypertext markup*: il markup delle pagine è semplice ed essenziale;
- *WikiPagesTitlesAreMashedTogether*: I link del wiki vengono creati unendo insieme due stringhe con la prima lettera maiuscola, ad esempio ‘HomePage’;

- *Content is ego-less, time-less, and never finished*: l'anonimità non è obbligatoria ma è comune, le pagine non sono organizzate cronologicamente, e nuovi contributi sono frequenti e ben accetti.

Un esempio famoso d'uso di questa tecnologia è la Wikipedia[26], un'enciclopedia ad accesso libero, che è senza dubbio il più grosso progetto wiki sul web. La versione inglese, online dal 2001, conta oggi circa 800 mila articoli.

#### *2.1.2.5 Calendari e schedulazione*

Questo tipo di tecnologie groupware includono sia funzioni di gestione del calendario, ovvero la possibilità di inserire dati all'interno di un calendario elettronico, sia di schedulazione, quindi la generazione automatica della data e dell'ora più convenienti per un incontro.
A questo scopo viene creato un diario condiviso tra gli utenti, accessibile via rete, su cui sofisticati programmi di schedulazione trovano il momento migliore per un meeting tra determinati partecipanti in base alla disponibilità dei membri del gruppo. La conferma eventualmente viene notificata tramite posta elettronica.

#### *2.1.2.6 MMORPG*

Alcuni tra i videogiochi più vecchi e famosi sono i giochi di ruolo[Bub02] (*role-playing games*, RPGs), come ad esempio *Dungeons & Dragons*.
Il concetto essenziale di questo tipo di giochi è che il giocatore impersona uno o più personaggi, usandoli per interagire con l'ambiente di gioco. Nel processo che porta al completamento della missione ('*quests*'), il giocatore può manipolare oggetti, conversare con altri personaggi e combattere con i mostri del mondo virtuale.
Nel momento in cui la possibilità di collegarsi alla rete divenne più "popolare" iniziarono ad apparire i primi giochi online text-based. Uno di questi, multi-user dungeons (MUDs), permetteva ai giocatori di interagire con gli altri e giocare ad un RPG in un ambiente virtuale condiviso [Cuc02].

Successivamente *Ultima Online* [Bub02] sviluppò questo concetto introducendo il primo famoso gioco di ruolo online grafico.
Un gran numero di giochi si sono affiancati a questo nuovo genere, creando il genere oggi conosciuto come *massively-multiplayer online role-playing games* (MMORPGs) che unisce le caratteristiche dei vari generi precedenti offrendo ai partecipanti un ambiente virtuale ricco di funzionalità.
La collaborazione è fondamentale in questo tipo di giochi, in cui a seconda del personaggio interpretato, si svolge una diversa funzione all'interno del mondo virtuale.

#### *2.1.2.7 Giochi online*

Esistono in rete un gran numero di giochi con scopi, regole e funzionalità molto diverse. I giochi che consideriamo sono quelli in cui possono partecipare due o più giocatori. I partecipanti possono alternarsi nelle mosse o giocare contemporaneamente, influenzandosi a vicenda. Il primo caso è quello dei giochi come la dama o la battaglia navale in cui ogni giocatore compie la propria mossa e poi aspetta che ogni avversario faccia lo stesso per riavere il turno. Il secondo caso è quello dei giochi in cui le mosse che vanno a buon fine da una parte aggiungono difficoltà dall'altra. Si consideri, ad esempio, una partita a tetris in cui due partecipanti si sfidano: ogni volta che un giocatore riesce a completare una linea, i pezzi dell'avversario scendono più velocemente.
Molti di questi giochi non solo permettono di giocare, ma anche di mandarsi messaggi testuali sia privatamente che pubblicamente.

#### *2.1.2.8 Desktop Sharing*

La tecnologia del desktop sharing permette che l'intero contenuto della finestra di un desktop possa essere visualizzato, così come si presenta sul proprio pc, da un gruppo di partecipanti collocati in luoghi diversi. Il controllo, solitamente, è mantenuto da un solo partecipante alla volta ma tutti possono vedere la stessa interfaccia grafica condivisa. A causa dell'impredicibilità dei cambiamenti nella finestra principale, l'immagine del desktop deve essere trasmessa periodicamente agli utenti del gruppo. Per questo motivo

il desktop sharing ha bisogno di una ragionevole quantità di banda per poter funzionare in modo apprezzabile.
Oltre a visualizzare il desktop condiviso, in alcuni casi, i partecipanti possono interagire con la finestra contemporaneamente, ad esempio, se si muove il mouse all'interno della finestra condivisa sul client, il puntatore sul server, e di conseguenza nella finestra di tutti gli altri partecipanti, si muoverà coerentemente.
Un esempio famoso di software per il desktop sharing è VNC (Virtual Network Computing)[VNC]. VNC è essenzialmente un sistema open source che permette di visualizzare uno schermo non solo sulla macchina cui appartiene ma su qualunque computer connesso alla rete Internet.
Poiché VNC è uscito con una licenza che permette agli altri sviluppatori di eseguire modifiche, è possibile scegliere tra un vasto assortimento di soluzioni basate su VNC.
In alcune implementazioni, VNC è ottimizzato per il controllo a distanza delle macchine, funzione molto utile per le applicazioni di supporto tecnico, di servizio clienti e di risoluzione dei problemi. In altre implementazioni, invece, può essere usato per il trasferimento di documenti.
Una soluzione non ancora famosa ma molto interessante è Glance Networks[Gla] un software che permette di mettere il proprio computer in modalità di condivisione dello schermo con un semplice tasto. Una volta attivata la modalità condivisione si può collaborare semplicemente comunicando l'url privato della pagina web di Glance agli altri partecipanti che potranno caricare sul proprio browser quella pagina e vedere esattamente cosa succede sullo schermo.
Esplorando più attentamente il sito di Glance Networks si trovano anche le seguenti informazioni: "Il software della Società Glance Networks include ed esegue una versione modificata di Virtual Network Computing (VNC) e passa messaggi da e per il motore modificato di VNC. Il codice sorgente della versione di VNC modificata da Glance si può scaricare gratuitamente".

#### *2.1.2.9 Workflow system*

Organizzare il lavoro e distribuire i compiti in funzione delle capacità e delle competenze di coloro che li svolgono è una pratica molto antica.

L' organizzatore è spesso colui che svolge questa attività di assegnazione, controlla lo stato di avanzamento dei lavori rispetto all'obiettivo finale e misura l'efficienza globale.
I sistemi di gestione dei flussi di lavoro (WFMS, *Workflow management Systems*) sono dei software di supporto per il lavoro collaborativo. Allen[All01] definisce i workflow come: "L'automazione di tutto o parte un processo di business, durante il quale documenti, informazioni o attività sono passati da un partecipante all'altro per essere elaborati in accordo con un insieme di regole procedurali".
Il sistema è tipicamente composto da un sottosistema che permette di definire l'organizzazione del lavoro (definizione del processo di lavoro e assegnazione dei compiti) e da un motore che la automatizza. In generale un sistema di Workflow offre vantaggi quali:

- Migliore monitoraggio e controllo dei processi;
- Minori costi di gestione dei processi;
- Migliore qualità di servizio
- Minori costi di addestramento del personale

Il concetto di workflow è strettamente legato alla realizzazione di groupware, considerando il lavoro di persone svolto attraverso l'uso dei computer.

In questo capitolo sono state analizzate le tecnologie per la collaborazione e sono state messe in evidenza le loro caratteristiche.
Nel prossimo capitolo si analizzeranno invece le soluzioni per la negoziazione, il formato dei messaggi scambiati tra le parti e i meccanismi di sicurezza che permettono di garantire confidenzialità, autenticazione e integrità dei dati trasmessi.

# Capitolo 3: Negoziazione, formato dei messaggi e sicurezza

In questo capitolo saranno descritti alcuni protocolli che ci sono stati utili per capire come affrontare i problemi relativi alla negoziazione. Inoltre verrà specificato il formato dei messaggi che vengono scambiati tra i partecipanti e verranno analizzati i meccanismi di sicurezza attualmente disponibili.

## 3.1 Content Negotiation e Device Independence

La negoziazione del contenuto è un argomento molto importante di cui tener conto quando, come nel nostro caso, si parla di entità diverse che vogliono scambiare informazioni (comunicare) e mettersi d'accordo in base alle proprie capacità.
Nel contesto della negoziazione le parti che vogliono collaborare spesso hanno a disposizione hardware e software di diversi produttori e per questo i documenti di una parte potrebbero non essere leggibili o visualizzabili dall'altra parte.
Per fare un esempio si consideri un sito web ottimizzato per schermi con risoluzione 800x600. Il contenuto di tale sito dovrebbe poter essere accessibile a chiunque ma in alcuni casi potrebbe non essere così. Supponiamo che un utente voglia accedere al sito tramite il suo smartphone, cosa succede a questo punto? Se il progettista è stato accorto, i contenuti del sito potrebbero essere fruibili in modo adatto, altrimenti in caso contrario potrebbero essere illeggibili.

La dimensione dello schermo non riguarda il contenuto del sito in sé ma è una caratteristica da tenere in considerazione se si vuole che molti utenti abbiano la possibilità di accedere ai contenuti senza difficoltà.
Il ramo che si occupa delle problematiche relative alla presentazione dei documenti in device diversi viene chiamato “Device Independence”[Gim03] .
Il semplice scenario presentato poco fa non riguarda da vicino il nostro lavoro ma alcune problematiche sono comuni; ci si occupa, infatti, della possibilità che le parti possano accordarsi sulle proprie capacità.
Nel seguito analizzeremo il funzionamento della negoziazione nel protocollo HTTP, poi ci sposteremo su una sua variazione, il Transparent Content Negotiation (TCN), proseguiremo con la descrizione del protocollo CC/PP (Composite Capability/Preference Profile), ed infine analizzeremo ebXML. I primi tre approcci analizzati sono protocolli e tecnologie nate nel contesto del “Device Independence” perciò riguardano in particolar modo il problema di negoziare le caratteristiche delle parti per poter adattare il contenuto web non solo per quanto riguarda l’architettura hardware e software ma anche per quanto riguarda le preferenze dell’utente. ebXML invece è uno standard per l’e-commerce che si propone di fornire modalità d’interazione tra “business information system” .

### 3.1.1 Negoziazione in HTTP

Il protocollo HTTP[FGMFB97] fornisce diversi meccanismi per la negoziazione del contenuto ossia il processo di selezione della miglior rappresentazione possibile di una HTTP-response laddove ve ne siano più d’una disponibili.
Esistono due tipi di negoziazione definiti dal protocollo: negoziazione server-driven e negoziazione agent-driven.
Questi due tipi sono ortogonali e possono essere usati separatamente o in combinazione. Un metodo di combinazione, il TCN, sarà descritto nel prossimo paragrafo.

Nella negoziazione server-driven la selezione della miglior rappresentazione della risposta è ottenuta mediante un algoritmo residente sul server. Ogni volta che il server riceve una richiesta analizza i valori degli attributi opzionali Accept presenti nell’header

e seleziona la rappresentazione più adatta. Naturalmente affinché l'algoritmo funzioni è necessario che i campi opzionali siano utilizzati.
Questa soluzione sfrutta gli header della richiesta per poter inviare immediatamente una risposta che suppone essere la migliore. Non è detto che lo sia poiché è possibile specificare con gli attributi Accept un limitato numero di caratteristiche che non sono sufficienti per descrivere le diverse funzionalità dei device o le preferenze dell'utente.
Alcuni campi Accept dell'header messi a disposizione dal protocollo HTTP sono, ad esempio, Accept, Accept-charset, Accept-Encoding, Accept-Language ma possono essere fornite anche informazione sullo user agent come il nome, il numero di versione o il produttore. Ecco un esempio:

```
accept-language it,it-it;q=0.8,en-us;q=0.5,en;q=0.3
accept-encoding gzip, defilate
accept text/xml, application/xml, application/xhtml+xml, text/html;
q=0.9, text/plain; q=0.8,image/png,*/*; q=0.5
accept-charset ISO-8859-1,utf-8;q=0.7,*;q=0.7
user-agent Mozilla/5.0 (Windows; U; Windows NT 5.1; it-IT; rv:1.7.10)
Gecko/20050717 Firefox/1.0.6
```

Nella negoziazione agent-driven il client invia una normale richiesta e riceve dal server una risposta con una lista delle rappresentazioni disponibili. La scelta a questo punto può essere fatta automaticamente dallo user agent (se capace) o direttamente dall'utente attraverso un menù ipertestuale creato dinamicamente.
Lo svantaggio di questa soluzione è evidente; essa, infatti, ha bisogno di una seconda richiesta HTTP, da parte del client, che contiene la scelta.

### 3.1.2 TCN

Il Transparent Content Negotiation è un protocollo sperimentale presentato da Holtman e Mutz[HolMut98] nel 1998 che sfrutta in combinazione i due tipi di negoziazione offerti dall'HTTP, server-driven e agent-driven.
Il TCN elimina il bisogno di spedire un lungo elenco di Accept-header e permette un processo di selezione che porta sempre alla miglior variante della pagina richiesta o ad

un errore nel caso in cui lo user agent non sia in grado di presentare in modo adeguato nessuna delle varianti disponibili.

Nello schema di questa negoziazione il client spedisce una comune richiesta HTTP-GET e il server invia in risposta la lista delle varianti disponibili (*list-response*). A questo punto lo user agent effettua la scelta della miglior rappresentazione e la invia al server che risponde inviando la risorsa nella variante richiesta. In questo schema le capacità e le preferenze sono usate esclusivamente dallo user agent.

La prima risposta del server contiene la lista delle varianti e il contenuto html incluso viene utilizzato per dare la possibilità all'utente di fare la scelta manualmente.

```
HTTP/1.1 300 Multiple Choices
     Date: Tue, 11 Jan 2006 20:02:21 GMT
     TCN: list
     Alternates: {"paper.1" 0.9 {type text/html} {language en}},
                 {"paper.2" 0.7 {type text/html} {language fr}},
                 {"paper.3" 1.0 {type application/postscript}
                     {language en}}
     Vary: negotiate, accept, accept-language
     ETag: "blah;1234"
     Cache-control: max-age=86400
     Content-Type: text/html
     Content-Length: 227

     <h2>Multiple Choices:</h2>
     <ul>
     <li><a href=paper.1>HTML, English version</a>
     <li><a href=paper.2>HTML, French version</a>
     <li><a href=paper.3>Postscript, English version</a>
     </ul>
```

Il TCN propone due metodi per ottimizzare il processo di negoziazione: da una parte i proxy cache potrebbero mantenere sia le liste di varianti che le varianti stesse in modo da ridurre il tempo necessario sia per ricevere la lista sia per ottenere la variante desiderata; inoltre lo user agent potrebbe comunque mandare nella prima richiesta alcuni Accept-header che potrebbero contenere abbastanza informazioni da permettere al server di scegliere la miglior variante e restituirla direttamente senza bisogno che il client effettui una scelta.

Come nella negoziazione server-driven di HTTP la scelta dal lato server è eseguita da un algoritmo, in questo caso denominato "*remote variant selection algorithm*", ma solo se lo user agent lo permette esplicitamente.

Infine, i due miglioramenti, l'uso dei proxy e di un algoritmo di selezione basato sugli Accept-header, possono essere usati insieme; il proxy che mantiene la lista può, in alcuni casi, essere capace di scegliere la miglior variante al posto dello user agent. Quest'ultima soluzione permette non solo di minimizzare l'uso della banda ma anche di limitare i ritardi dovuti al trasferimento e di diminuire il carico di lavoro sul server.
Il TCN definisce quattro dimensioni per la negoziazione: Media type, Charset, Language, e Features.
Le prime tre dimensioni sono presenti in HTTP, la quarta invece è una novità introdotta dal protocollo TCN. All'interno della dimensione Features si possono negoziare caratteristiche come, ad esempio, estensioni HTML o estensioni di altri media type, dimensione dello schermo, colori, o dettagli grafici. L'introduzione di questa dimensione è il modo in cui il TCN offre l'estensibilità poiché è sempre possibile registrare una nuova caratteristica all'interno di Features.

### 3.1.3 CC/PP

Il 15 gennaio 2004 il World Wide Web Consortium (W3C[30]) annuncia la pubblicazione di Composite Capability/Preference Profiles: Structure and Vocabularies 1.0 Recommendation[KRWOHBT04]. CC/PP è un sistema per esprimere le capacità dei dispositivi e le preferenze dell'utente usando il Resource Description Framework (RDF). L' utilizzo di RDF per CC/PP ha molti vantaggi, tra cui:

- Vocabolari estensibili: Qualsiasi fornitore di dispositivi può definire un vocabolario che può essere facilmente riutilizzato ed esteso.
- Vocabolari non centralizzati: Chiunque può definire nuove capacità dei dispositivi che possono integrarsi agevolmente con altre già esistenti.

CC/PP permette descrizioni complesse e complete di tutti gli aspetti della distribuzione del contenuto e fornisce un'informazione completa per il processo di adattamento alle necessità dell'utente. Tutto ciò è disponibile attraverso la creazione di un profilo. Un profilo CC/PP è una collezione di attributi i cui valori possono essere utilizzati dal server per stabilire quale sia la risorsa più appropriata da inviare al client. Tale descrizione è collocata in rete e possiede un identificatore univoco (URI).

Il profilo è strutturato in modo tale da permettere al client e ad un eventuale proxy di descrivere le loro caratteristiche facendo riferimento ad un profilo "standard", accessibile dal lato server o da un altro fornitore di servizi, specificando solo le caratteristiche che vanno aggiunte poiché mancanti o modificate in quanto differenti da quelle standard.

Un profilo CC/PP è quindi una descrizione delle capacità di un dispositivo e delle preferenze associate all'utente che lo utilizza che possono essere utilizzate per accedere ad una risorsa Web adattandone i contenuti in base alle caratteristiche specificate. Esse includono la piattaforma Hardware, il sistema operativo e le applicazioni usate dall'utente. Ecco un esempio di profilo CC/PP:

```
<?xml version="1.0"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
         xmlns:ex="http://www.example.com/schema#">

  <rdf:Description
      rdf:about="http://www.example.com/profile#MyProfile">

    <ccpp:component>
      <rdf:Description
          rdf:about="http://www.example.com/profile#TerminalHardware">
        <rdf:type
rdf:resource="http://www.example.com/schema#HardwarePlatform" />
        <ex:displayWidth>320</ex:displayWidth>
        <ex:displayHeight>200</ex:displayHeight>
      </rdf:Description>
    </ccpp:component>

    <ccpp:component>
      <rdf:Description
          rdf:about="http://www.example.com/profile#TerminalSoftware">
        <rdf:type
rdf:resource="http://www.example.com/schema#SoftwarePlatform" />
        <ex:name>EPOC</ex:name>
        <ex:version>2.0</ex:version>
        <ex:vendor>Symbian</ex:vendor>
      </rdf:Description>
    </ccpp:component>

    <ccpp:component>
      <rdf:Description
          rdf:about="http://www.example.com/profile#TerminalBrowser">
        <rdf:type
            rdf:resource="http://www.example.com/schema#BrowserUA" />
        <ex:name>Mozilla</ex:name>
        <ex:version>5.0</ex:version>
        <ex:vendor>Symbian</ex:vendor>
        <ex:htmlVersionsSupported>
```

```
          <rdf:Bag>
            <rdf:li>3.2</rdf:li>
            <rdf:li>4.0</rdf:li>
          </rdf:Bag>
        </ex:htmlVersionsSupported>
      </rdf:Description>
    </ccpp:component>

  </rdf:Description>
</rdf:RDF>
```

### 3.1.4 ebXML

ebXML[Ais02], proposto da OASIS e UN/CEFAT, permette che due entità si possano mettere d'accordo su vari parametri per iniziare una *"Business Collaboration"*. In ebXML lo scambio di dati fra le due parti richiede che ciascuna parte sia a conoscenza delle Business Collaboration supportate dall'altra parte, il ruolo da questa ricoperto ed i dettagli tecnologici circa il modo in cui invia e riceve i messaggi.

Per facilitare questo scambio d'informazioni, ebXML descrive il Collaboration Protocol Profile (CPP) ed il Collaboration Protocol Agreement (CPA).

Il Collaboration Protocol Profile (CPP) definisce le potenzialità ed i modi in cui una parte può impegnarsi in business elettronico con altre parti. Queste potenzialità sono sia tecnologiche sia commerciali. Una parte può descrivere se stessa in un unico CPP, oppure, può creare differenti CPP che descrivono le diverse Business Collaborations supportate.

La struttura generale di un CPP è la seguente:

```
<tp:CollaborationProtocolProfile
  xmlns:tp=
"http://www.oasis-open.org/committees/ebxmlcppa/schema/cpp-cpa-_0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
"http://www.oasis-open.org/committees/ebxmlcppa/schema/cpp-cpa-2_0.xsd
http://www.oasis-open.org/committees/ebxmlcppa/schema/cpp-cpa-2_0.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  tp:cppid="uri:companyA-cpp"
  tp:version="2_0b">
      <tp:PartyInfo> <!-- one or more -->
      ...
      </tp:PartyInfo>
      <tp:SimplePart id="..."> <!-- one or more -->
      ...
      </tp:SimplePart>
      <tp:Packaging id="..."> <!-- one or more -->
      ...
      </tp:Packaging>
```

```
      <tp:Signature> <!-- zero or one -->
      ...
      </tp:Signature>
      <tp:Comment>text</tp:Comment> <!-- zero or more -->
      </tp:CollaborationProtocolProfile>
```

Per facilitare la ricerca di possibili Business Partner, i CPP possono essere immagazzinati in un apposito repository pubblico, ad esempio, l'ebXML Registry (ebRS).

Il Collaboration Protocol Agreement (CPA) definisce le potenzialità messe a disposizione dalle due parti sulle quali esse devono trovarsi in accordo prima di impegnarsi nelle transazioni descritte dal CPA. L'informazione di un CPA è usata per configurare i sistemi dei due Partner commerciali per rendere possibile lo scambio di messaggi durante lo svolgimento delle Business Collaboration selezionate. Il CPA è creato mediante calcoli e negoziazioni derivanti dall'incrocio di due CPP.

Ecco la struttura di un CPA:

```
<CollaborationProtocolAgreement
      xmlns:tp=
"http://www.oasis-open.org/committees/ebxmlcppa/schema/cpp-cpa-_0.xsd"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      xmlns:xlink="http://www.w3.org/1999/xlink"
      tp:cpaid="YoursAndMyCPA"
      tp:version="2.0a">
      <tp:Status tp:value="proposed"/>
      <tp:Start>1988-04-07T18:39:09</Start>
      <tp:End>1990-04-07T18:40:00</End>
      <!-- ConversationConstraints MAY appear 0 or 1 time -->
      <tp:ConversationConstraints
      tp:invocationLimit="100"
      tp:concurrentConversations="4"/>
      <tp:PartyInfo>
      ...
      </tp:PartyInfo>
      <tp:PartyInfo>
      ...
      </tp:PartyInfo>
      <tp:SimplePart tp:id="..."> <!-- one or more -->
      ...
      </tp:SimplePart>
      <tp:Packaging tp:id="..."> <!-- one or more -->
      ...
      </tp:Packaging>
      <tp:Signature> <!-- zero or one time -->
      ...
      </tp:Signature>
      <tp:Comment>any text</tp:Comment>  <!--zero or more -->
      </tp:CollaborationProtocolAgreement>
```

Il processo di trasmissione dei messaggi viene effettuato usando il servizio di messaggistica di ebXML, definito sulla base di pacchetti SOAP.

## 3.2 Messaggi e SOAP

Il Simple Object Access Protocol (SOAP)[Mit03], [GHMMN03a], [GHMMN03b] è un protocollo basato su XML che consente a due applicazioni di comunicare tra loro sul Web. Pubblicato come Working Draft dal W3C nel dicembre 2001, dal 2003 SOAP è una Recommendation che definisce il formato dei messaggi che due applicazioni possono scambiarsi utilizzando i protocolli Internet, come ad esempio HTTP o SMTP, per fornire dati e richiedere elaborazioni. Il protocollo è indipendente dalla piattaforma hardware e software ed è indipendente dal linguaggio di programmazione utilizzato per sviluppare le applicazioni.
Grazie a questo protocollo, è possibile affiancare alle informazioni per la pubblicazione classica delle pagine Web, informazioni destinate alle applicazioni.

### 3.2.1 Formato dei messaggi SOAP

Un messaggio SOAP è un documento XML che descrive una richiesta di elaborazione o il risultato di una elaborazione. In particolare, un messaggio SOAP è costituito dai seguenti elementi:

- **Envelope:** Rappresenta il contenitore del messaggio e costituisce l'elemento radice del documento XML
- **Header:** È un elemento opzionale che contiene informazioni globali sul messaggio;
- **Body:** all'interno del body si trova il contenuto vero e proprio;
- **Fault:** Se presente, fornisce informazioni sugli errori che si sono verificati durante l'elaborazione; quest'elemento può essere presente soltanto nei messaggi di risposta.

I messaggi scambiati tra le parti, siano questi messaggi che trasportano il contenuto vero e proprio o messaggi con cui le parti si accordano sulle capacità collaborative, saranno incapsulati in messaggi SOAP.

## 3.3 Sicurezza

Le parti che vogliono iniziare una collaborazione potrebbero avere bisogno di meccanismi di sicurezza sia per quanto riguarda la protezione dei messaggi sia per quanto riguarda l'identificazione dei partecipanti.
I messaggi possono essere firmati, criptati, o entrambi e i partecipanti possono dover esibire un certificato che attesti la propria identità.
Queste funzionalità sono fornite attraverso l'uso dei meccanismi descritti dai protocolli XML Signature e XML Digital Encryption.

### 3.3.1 XML Signature

XML Signature[BBFLS02] permette di applicare una firma digitale per garantire confidenzialità e integrità ai documenti XML trasmessi.
XML consente di esprimere le stesse informazioni in modi diversi, cosicché due documenti XML possono essere logicamente equivalenti ma possono avere differenze, come gli spazi all'interno dei tag o i commenti, che non riguardano la loro struttura logica. Nel firmare dati XML bisogna tenere in mente questa flessibilità del linguaggio, infatti, nonostante le applicazioni XML possano ignorare quelle differenze perché non hanno impatto sull'informazione in sé, tutto ciò è molto importante per le firme digitali perché la validazione deve essere applicata esattamente allo stesso flusso di byte di quando è stata generata.
Per risolvere questo problema è stato introdotto il concetto di canonizzazione (canonicalization) dei documenti XML[Boy01]. Convertire un documento XML nella sua forma canonica significa applicargli un insieme di trasformazioni in modo da rappresentare le informazioni variabili in maniera standard. Così, due documenti logicamente equivalenti avranno esattamente la stessa forma canonica.

Per creare una signature XML per un qualsiasi contenuto digitale si deve, per prima cosa, ottenere un valore "digest" per ciascun oggetto (dato) che si vuole firmare. L'insieme risultante di questi valori viene inserito in un elemento XML insieme ad altre informazioni; viene generato un valore "digest" dell'elemento appena descritto e viene firmato con una chiave privata. Il processo di validazione, quindi, prevede due passi: prima viene validata la firma stessa assicurando l'integrità del valore "digest" dell'intero elemento firmato, poi è validato il valore "digest" di ciascun dato.

L'elemento Signature che rappresenta la firma digitale ha la seguente struttura. Si userà "?" per indicare zero o un'occorrenza di un elemento, "+" per indicare una o più occorrenze e "*" per indicare zero o più occorrenze.

```
<Signature ID?>
      <SignedInfo>
            <CanonicalizationMethod/>
            <SignatureMethod/>
            (<Reference URI?>
                  (<Tranforms>)?
                  <DigestMethod/>
                  <DigestValue/>
             </Reference>)+
      </SignedInfo>
      <SignatureValue>...</SignatureValue>
      (<KeyInfo>)?
      (<Object ID?>)*
</Signature>
```

L'elemento Signature contiene quattro sotto-elementi: SignedInfo, SignatureValue, KeyInfo ed Object. L'elemento SignedInfo rappresenta le informazioni realmente firmate. Questo elemento contiene anche informazioni sul metodo di canonizzazione e sull'algoritmo di firma usati per calcolare il SignatureValue.
SignedInfo contiene un elemento Reference per ciascun oggetto firmato che include un riferimento all'oggetto (attraverso un URI), il metodo usato per calcolare il valore "digest" dello stesso ed il corrispondente valore "digest". Un elemento Reference può anche contenere delle trasformazioni che, se applicate all'oggetto identificato,

producono l'input per l'operazione di calcolo del valore "digest" (un esempio di tali trasformazioni è il metodo di canonizzazione spiegato in precedenza).
L'elemento KeyInfo indica la chiave da usare per generare (e validare) la firma. Può includere le chiavi, i nomi delle chiavi, i certificati ed ogni altra informazione utile per il processo di crittografia. Infine, l'elemento Object viene usato per includere qualsiasi altro oggetto all'interno dell'elemento Signature e può essere firmato o meno.

### 3.3.2 XML Digital Encryption

La codifica tramite XML Digital Encryption[IDS02] può essere applicata sia ad interi documenti XML che ad elementi. Dopo un processo di XML encryption si ottiene un elemento EncryptedData che contiene, tramite un sotto-elemento o un riferimento, i dati da cifrare. Se questi dati costituiscono un elemento XML, EncryptedData sostituirà quest'ultimo nella versione cifrata del documento XML finale.
L'elemento EncryptedData ha la seguente struttura:

```
<EncryptedData Id? Type? MimeType? Encoding?>
        <EncryptionMethod/>?
        <ds:KeyInfo>
                <EncryptedKey>?
                <AgreementMethod>?
                <ds:KeyName>?
                <ds:RetrievalMethod>?
                <ds:*>?
        </ds:KeyInfo>?
        <CipherData>
                <CipherValue>?
                <CipherReference URI?>?
        </CipherData>
        <EncryptionProperties>?
</EncryptedData>
```

EncryptedData contiene quattro elementi principali: il primo di essi, EncryptionMethod, indica l'algoritmo di cifratura usato, mentre il secondo, ds:KeyInfo, identifica la chiave usata per cifrare i dati. CipherData contiene i dati cifrati o un riferimento alla loro

posizione, infine, l'elemento opzionale EncryptionProperties fornisce altre informazioni sulla generazione dei dati cifrati, ad esempio un timestamp.

Gli elementi EncryptedMethod e ds:KeyInfo sono opzionali in quanto mittente e destinatario potrebbero essersi accordati in anticipo sull'algoritmo di cifratura e sulla chiave da usare. L'elemento ds:KeyInfo offre modi differenti di fornire la chiave sia cifrata in un elemento EncryptedKey sia attraverso un sistema di gestione delle chiavi, tramite un elemento AgreementMethod; oppure fornendo l'identificativo di una chiave nota nell'elemento ds:KeyName o il riferimento alla posizione dove la chiave è memorizzata tramite l'elemento ds:RetrievalMethod.

# Capitolo 4: Un nuovo modello

Siamo convinti che ogni collaborazione possa essere descritta tramite 5 caratteristiche che la distinguono. In questo capitolo analizzeremo in dettaglio quali siano queste caratteristiche.
Una volta definito lo spazio della collaborazione si passerà all'analisi vera e propria delle tecnologie collaborative.

## 4.1 Caratteristiche della collaborazione

Le tecnologie collaborative analizzate nel Capitolo 2 saranno ora classificate secondo criteri che descrivono sotto ogni aspetto le loro caratteristiche.
Questa classificazione permette di distinguere le varie tecnologie e di individuare quali possono interagire tra loro grazie alle caratteristiche comuni. I criteri analizzati descrivono completamente la conversazione e il tipo di collaborazione che la conversazione rende disponibile e, in particolare, ogni conversazione è identificata dal rapporto con la risorsa, dal meccanismo di turni, dal formato dati delle espressioni, dalla visibilità e dal protocollo d'aggiornamento.
Ogni dimensione, secondo le capacità di ciascuna tecnologia, potrà assumere un valore a scelta tra quelli disponibili e la sequenza di tutti i valori delle dimensioni identificherà univocamente un determinato tipo di conversazione.

### 4.1.1 Il rapporto con la risorsa

La prima dimensione che analizziamo è il rapporto con la risorsa. In alcune tecnologie, come ad esempio nel disegno condiviso, la collaborazione avviene modificando una risorsa condivisa tra i partecipanti; altre tecnologie, come le chat, non hanno una risorsa che viene alterata durante una conversazione. Oltre a questi due casi è anche possibile che esista una risorsa ma questa non venga modificata durante la collaborazione, come nel caso delle annotazioni in cui i collaboratori compiono operazioni, come aggiungere o modificare commenti esterni, solo riferendosi alla risorsa.
Il primo criterio, dunque, ci permette di analizzare come la risorsa influisce sullo stile della collaborazione, in altre parole come cambia, secondo la tecnologia utilizzata, la relazione tra i collaboratori e la risorsa, se esiste, su cui essi collaborano.
Distinguiamo tre tipi di relazione:

- La collaborazione CIRCA una risorsa
- La collaborazione SU una risorsa
- La collaborazione che E' la risorsa

Nel primo caso i collaboratori condividono una risorsa ma questa non viene modificata.
Le espressioni scambiate non modificano il contenuto ma possono fare riferimento ad esso o ad alcune parti. Un esempio sono i link ed i commenti esterni.
La risorsa viene utilizzata come punto di riferimento su cui vengono scambiate informazioni aggiuntive o commenti che possono essere mantenuti su un server secondario capace di metterli a disposizione su richiesta.
Dato che le informazioni sono secondarie e non modificano la risorsa originale, non ci sono rischi di cancellazioni accidentali o cambiamenti non autorizzati quindi non c'è necessità di meccanismi di controllo della consistenza del documento o di controllo d'accesso.

Nel secondo caso la risorsa viene modificata durante la collaborazione.

Sono un esempio gli editor condivisi in cui la risorsa comune a tutti i partecipanti assume stati diversi man mano che si aggiunge nuovo contenuto e si apportano modifiche.
In questi casi ha un'importanza vitale il meccanismo di controllo di consistenza che preserva l'ordine delle espressioni.
E' altresì importante un meccanismo di blocco del documento quando un partecipante compie delle modifiche. Il blocco può essere totale, in altre parole sull'intera risorsa, o parziale e in questo caso il documento è considerato come composto da sezioni e solo quelle occupate vengono bloccate, lasciando gli altri collaboratori liberi di lavorare contemporaneamente sulle altre parti.
Può spesso essere necessario dover eseguire controlli d'accesso, nel caso in cui si voglia assicurare una gestione regolare dell'evoluzione del contenuto della risorsa.

Quando la collaborazione E' la risorsa, non esiste una risorsa vera e propria a cui associare le espressioni ma sono le espressioni che la creano.
La collaborazione non ha come scopo principale la creazione di una risorsa (anche se in alcuni casi è possibile salvare il risultato di una collaborazione affinché diventi una risorsa vera e propria) ma la conversazione e la collaborazione sono, in effetti, la stessa cosa. Questo non significa che non possa esistere una risorsa ma semplicemente che lo stato di questa non ha importanza ai fini della collaborazione. Si consideri per fare un esempio un gioco online in cui i partecipanti muovono le proprie pedine su una scacchiera condivisa. Senza dubbio la collaborazione avviene sulla risorsa-scacchiera ma lo stato che essa assume durante la collaborazione non ha importanza ai fini del gioco. Quello che importa è il risultato della partita. Ci riferiremo a questo tipo di risorse come *risorse temporanee*.
Appartengono a questa categoria le chat, i giochi online, le whiteboard condivise.
Alcune applicazioni possono dover mettere a disposizione un rigoroso meccanismo di turno (per intenderci come negli scacchi), altre possono essere più flessibili, come ad esempio un forum.

La dimensione **Rapporto con la risorsa**, riassumendo, può assumere tre diversi valori: nel caso di collaborazione su una risorsa il valore è **`onResource`**, nel caso di

collaborazione circa una risorsa il valore è **`aboutResource`**, e nel terzo caso, con una collaborazione che è la risorsa il valore associato è **`isResource`**.

### 4.1.2 Il meccanismo di turno

Come descritto nel Capitolo 2, le tecnologie CSCW sono classificate in una matrice bidimensionale di spazio e tempo. Da una parte lo spazio permette di distinguere applicazioni locali e remote, dall'altra il tempo, applicazioni sincrone e asincrone.
Per gli scopi di questo progetto non ci si riferirà al concetto di sincrono/asincrono in sé, ma si analizzerà il set di regole per guadagnare o perdere il diritto di produrre un'espressione in qualunque momento della conversazione.
Si seguirà un modello basato sui turni in cui ogni partecipante potrà contribuire alla collaborazione in modi diversi secondo il tipo di rapporto collaborativo che si vuole creare.
In alcuni casi è necessario che le espressioni vengano elaborate in un ordine rigoroso che implica un controllo su chi in ogni momento è autorizzato ad inviare espressioni. Alcune tecnologie limitano le modifiche ad un solo partecipante alla volta, altre invece eseguono controlli sulle diverse sezioni del documento permettendo che partecipanti diversi possano modificare contemporaneamente parti diverse. Oltre a questi casi, è possibile che i partecipanti siano, invece, liberi di inviare una o più espressioni in qualunque momento, in pratica, senza nessun meccanismo di turno né di blocco del documento o di alcune parti.
Abbiamo dunque riscontrato la necessità di due parametri che caratterizzano il turno: esclusività e numerosità.

#### *4.1.2.1 Esclusività*

Ci si chiede se durante la conversazione uno, alcuni o tutti i partecipanti hanno il diritto di produrre un'espressione e condividerla. In pratica questo parametro stabilisce quale tipo di controllo venga effettuato sull'invio di espressioni da parte dei partecipanti.
Si mettono a disposizione tre valori possibili:

- **`noLock`**: non ci sono vincoli per fare modifiche. Nel caso esista una risorsa sulla quale si collabora, ogni partecipante può liberamente cambiare il contenuto senza preoccuparsi se altri stanno facendo lo stesso. Nel caso di risorse temporanee o di collaborazioni aboutResource chiunque può inviare espressioni contemporaneamente, come nei newsgroup o nei forum.
- **`localLock`**: i partecipanti possono lavorare contemporaneamente ma il blocco della risorsa vera e propria o temporanea si limita alle parti che sono effettivamente sotto modifica. La dimensione delle sezioni in cui dividere la risorsa è arbitraria e dipende dal formato dati e dalla tecnologia utilizzata. Ad esempio, nel caso di editor condivisi potrebbe essere una frase, una pagina o un capitolo. Per specificare questa caratteristica utilizzeremo il campo `grain`.
- **`globalLock`**: in questo caso solo un partecipante per volta può produrre espressioni e nessun altro ha la possibilità di inviare espressioni finché il proprietario del turno non rilascia il lock.

Si noti che nel caso di collaborazioni `aboutResource` l'unico valore dell'esclusività è `noLock` . Questo dipende dal fatto che qualunque conversazione che fa riferimento ad una risorsa permette per sua natura il libero invio di espressioni, poiché queste, in qualunque numero, di qualunque tipo e provenienti da qualunque partecipante non si influenzano tra loro né modificano la risorsa a cui si riferiscono.

#### *4.1.2.2 Numerosità*

Nel caso di un lock (`localLock` e `globalLock`) alla parte con il turno è permesso produrre una sola espressione o può produrne diverse?
Se ogni partecipante ha al massimo un'espressione a disposizione per ogni turno significa che il lock viene rilasciato automaticamente dopo la condivisione dell'espressione mentre nel caso il proprietario del turno possa produrre svariate espressioni potrebbe anche dover rilasciare manualmente il lock quando ha finito.
I valori possibili sono dunque:

- **`singleUtterance`**: una espressione per turno;

- **`multipleUtterance`**: più espressioni per turno.

Nel caso di `noLock` la numerosità è sempre `multipleUtterance` poiché l'assenza di un meccanismo di blocco permette che ogni partecipante possa inviare quante espressioni desidera.
Per lo stesso motivo, nel caso di `aboutResource` l'unico valore possibile è `multipleUtterance`. Anche nel caso di una conversazione `onResource` l'unico valore disponibile è `multipleUtterance` poiché sia con `localLock` che con `globalLock` non ha senso parlare di `singleUtterance`. Infatti, se il partecipante ha un lock su tutta la risorsa allora può apportare tutte le modifiche che vuole in tutto il documento o se ha un lock locale solo su una parte, in ogni caso, non ci sono limiti al numero di espressioni che può produrre.

### 4.1.3 Il formato dati

Il formato dati di ogni espressione scambiata in una conversazione è molto importante per determinare il tipo di applicazione collaborativa.
Ad esempio una chat o un forum permetterà di usare testo semplice (o in alcuni casi html) mentre il disegno condiviso utilizzerà forme (magari elementi svg) o comandi di formattazione, un editor condiviso richiederà comandi di editing, i giochi mosse legali.
Molte tecnologie utilizzano lo stesso formato, altre permettono invece di scambiare più espressioni in formati diversi, ad esempio i quasi tutti i programmi di Instant messaging permettono non solo di scambiare testo ma consentono anche l'invio di file.
Prenderemo in considerazione i seguenti tipi di dato:

1. testo
2. audio
3. video
4. oggetto
5. comando

Nuovi formati potranno essere specificati attraverso il tipo MIME application/x-.

### 4.1.4 Visibilità

Alcune tecnologie prima di inviare un'espressione permettono all'utente di sapere chi la riceverà. In alcuni casi questo gruppo può essere limitato ad un certo numero di persone e potenzialmente crescere, ma sarà sempre possibile per un'utente che invia la sua espressione sapere chi saranno i destinatari.
E' il caso ad esempio delle email in cui so esattamente a chi invierò il mio messaggio, in particolare perché seleziono personalmente il o i destinatari; ma è anche il caso degli editor condivisi o dell'instant messaging.
In altri casi invece non ho modo di sapere chi potrà vedere il mio contributo, come nei forum, nei wiki, o nei MMORPG, perché, ad esempio, non posso sapere chi sta leggendo una pagina del wiki in cui ho inviato l'espressione o chi è collegato al server di gioco e vede o controlla le mie mosse.
I valori possibili per questa dimensione saranno **`public`** e **`private`** e indicheranno rispettivamente: le conversazioni in cui so chi vedrà il mio contributo e quelle in cui chiunque potrà vederlo.

### 4.1.5 Il protocollo di update

Le tecnologie collaborative si differenziano anche per il modo in cui ogni partecipante ottiene gli aggiornamenti.
Abbiamo evidenziato tre modi di essere aggiornati:

- **`pull`**: si declina all'utente il compito di richiedere gli aggiornamenti quando necessari.
- **`pseudoPush`** (o *polling*): anche in questo caso la richiesta arriva dalla parte del client e non è l'utente che provvede: è il sistema che ad ogni tot di tempo fissato dall'utente fa richiesta degli aggiornamenti.
- **`realPush`**: il client rimane in ascolto di messaggi d'aggiornamento che vengono inoltrati da un server o da un altro client non appena disponibili. Da parte sua il client in ascolto spedisce all'occorrenza i dati a sua disposizione.

## 4.2 Analisi delle tecnologie

Saranno analizzate ora le tecnologie descritte nel capitolo 2 in relazione alle dimensioni appena introdotte.

### 4.2.1 Email

L'email mette a disposizione due classi di conversazioni: da una parte abbiamo l'invio di testo, dall'altra l'invio di file.
Nell'analisi della tecnologia rispetto alle caratteristiche citate nel paragrafo 2.2, notiamo che in entrambe le conversazioni il rapporto con la risorsa è del terzo tipo (*isResource*) e non esiste alcun vincolo riguardo il meccanismo di turni. Chiunque può mandare un'email dovunque e in un qualsiasi momento. I valori dunque sono: noLock, multipleUtterance.
Nel caso della conversazione testuale il formato dati è testo mentre nella conversazione basata sugli allegati (si consideri l'invio di email senza testo) il formato dati è oggetto.
Generalmente il modello di aggiornamento è pull ma in alcuni casi, se i client di posta inviano richieste al server ad intervalli regolari, può essere pseudoPush.
Riassumendo, per la tecnologia email abbiamo i seguenti tipi di conversazione:

a) **Messaggi privati con ricezione su richiesta:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `testo`
   Visibilità: `private`
   Aggiornamento: `pull`

b) **Allegati privati con ricezione su richiesta:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`

Numerosità: `multipleUtterance`
Formato dati: `oggetto`
Visibilità: `private`
Aggiornamento: `pull`

**c) Messaggi privati con ricezione su richiesta temporizzata:**
Rapporto con la risorsa: `isResource`
Esclusività: `noLock`
Numerosità: `multipleUtterance`
Formato dati: `testo`
Visibilità: `private`
Aggiornamento: `pseudoPush`

**d) Allegati privati con ricezione su richiesta temporizzata:**
Rapporto con la risorsa: `isResource`
Esclusività: `noLock`
Numerosità: `multipleUtterance`
Formato dati: `oggetto`
Visibilità: `private`
Aggiornamento: `pseudoPush`

### 4.2.2 Forum e Newsgroup

I messaggi inviati per essere aggiunti ad un forum o ad un newsgroup si riferiscono ad una risorsa temporanea che può essere trasformata in una risorsa vera e propria salvando i messaggi. Non ci sono limitazioni per quanto riguarda i turni poiché ogni partecipante può liberamente inviare le proprie espressioni quando lo desidera.
Inoltre, dato che chiunque può accedere ad un forum e leggere i messaggi, la visibilità è public. Le conversazioni messe a disposizione sono due: l'invio di testo semplice e l'invio di oggetti.
Abbiamo i seguenti tipi di conversazione:

a) **Messaggi pubblici con ricezione su richiesta:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `testo`
   Visibilità: `public`
   Aggiornamento: `pull`

b) **Allegati pubblici con ricezione su richiesta:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `oggetto`
   Visibilità: `public`
   Aggiornamento: `pull`

### 4.2.3 Blog

Nell'analizzare le conversazioni disponibili attraverso la tecnologia dei blog consideriamo in primo luogo il contributo dell'autore. Egli fornisce le espressioni che sono la risorsa (il rapporto è dunque isResource) e non vi sono meccanismi di turno. La conversazione è pubblica poiché chiunque può collegarsi al sito e leggere il contenuto e il formato dati è testo.
In secondo luogo si analizza la conversazione dei visitatori che aggiungono commenti ai post dell'autore. Queste conversazioni si riferiscono ai post dell'autore, dunque sono conversazioni aboutResource, non ci sono meccanismi di turno (noLock), il formato dati è testo e la conversazione è pubblica. Infine gli aggiornamenti vengono visualizzati su richiesta in entrambi i casi quindi il meccanismo di update è pull.
Riassumendo, le conversazioni messe a disposizione dal blog sono:

a) **Messaggi pubblici con ricezione su richiesta:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `testo`
   Visibilità: `public`
   Aggiornamento: `pull`

b) **Commenti pubblici con ricezione su richiesta:**
   Rapporto con la risorsa: `aboutResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `testo`
   Visibilità: `public`
   Aggiornamento: `pull`

### 4.2.4 Chat

La chat mette a disposizione conversazioni in cui non esiste una risorsa vera e propria che viene modificata durante la collaborazione. Come abbiamo detto precedentemente, la conversazione è libera (*noLock*) e non ci sono vincoli sul numero di espressioni che ogni partecipante può produrre.
La chat permette di inviare sia testo che oggetti (qualunque tipo di file) e per quanto riguarda l'update è sempre il server ad informare l'utente dell'avvenuta produzione di una nuova espressione inviandola direttamente.
La chat mette a disposizione le seguenti conversazioni:

a) **Messaggi pubblici con ricezione automatica:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `public`

Aggiornamento: `realPush`

**b) Messaggi privati con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `private`

Aggiornamento: `realPush`

**c) Allegati pubblici con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `oggetto`

Visibilità: `public`

Aggiornamento: `realPush`

**d) Allegati privati con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `oggetto`

Visibilità: `private`

Aggiornamento: `realPush`

**e) Audio privato con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`
Formato dati: `audio`
Visibilità: `private`
Aggiornamento: `realPush`

**f) Video privato con ricezione automatica:**
Rapporto con la risorsa: `isResource`
Esclusività: `noLock`
Numerosità: `multipleUtterance`
Formato dati: `video`
Visibilità: `private`
Aggiornamento: `realPush`

### 4.2.5 Istant Messaging

A seconda del software utilizzato le funzionalità offerte dall'instant messaging possono essere varie. In generale sono state considerate tutte conversazioni possibili anche se non fornite da tutti i software.
Questa tecnologia offre la possibilità di inviare messaggi, oggetti, audio e video privatamente e senza limitazioni di esclusività o numerosità delle espressioni.
Le conversazioni fornibili dalla tecnologia IM sono dunque:

**a) Messaggi privati con ricezione automatica:**
Rapporto con la risorsa: `isResource`
Esclusività: `noLock`
Numerosità: `multipleUtterance`
Formato dati: `testo`
Visibilità: `private`
Aggiornamento: `realPush`

**b) Allegati privati con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `oggetto`

Visibilità: `private`

Aggiornamento: `realPush`

**c) Audio privato con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `audio`

Visibilità: `private`

Aggiornamento: `realPush`

**d) Video privato con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `video`

Visibilità: `private`

Aggiornamento: `realPush`

### 4.2.6 VoIP

Nel VoIP la relazione con la risorsa è del terzo tipo perché non esiste una risorsa che viene modificata con l'atto della collaborazione (*isResource)*. Come già detto, la tecnologia VoIP permette l'invio di audio, oggetti o testo e ogni partecipante può inviare espressioni quando vuole e in qualunque numero. A prescindere dal tipo di dato inviato tutte le conversazioni non hanno vincoli di turno.

Come nel caso dell'IM, in cui a seconda del software utilizzato le funzionalità a disposizione possono essere numerose o possono limitarsi al solo invio di testo, anche nel VoIP alcuni software permettono il solo scambio audio tra due persone.
Le conversazioni disponibili sono allora:

a) **Messaggi privati con ricezione automatica:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `testo`
   Visibilità: `private`
   Aggiornamento: `realPush`

b) **Allegati privati con ricezione automatica:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `oggetto`
   Visibilità: `private`
   Aggiornamento: `realPush`

c) **Audio privato con ricezione automatica:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `audio`
   Visibilità: `private`
   Aggiornamento: `realPush`

d) **Video privato con ricezione automatica:**
   Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `video`

Visibilità: `private`

Aggiornamento: `realPush`

### 4.2.7 Videoconferenza

Nella videoconferenza la comunicazione e la collaborazione sono la stessa cosa e non esiste una risorsa che viene modificata durante la sessione collaborativa. Il rapporto con la risorsa è, dunque, isResource.
Per quanto riguarda il meccanismo di turni i partecipanti non condividono un lock e non ci sono vincoli riguardo al numero di espressioni che ciascuno può produrre.
La modalità di aggiornamento infine è realPush perché i partecipanti ricevono i dati non appena sono disponibili senza doverli richiedere.
I tipi di dati che possono essere supportati dalle tecnologie di videoconferenza dipende dal software utilizzato. Come descritto nel paragrafo 2.1.1.7 alcuni software permettono oltre ai formati di base (audio e video) anche conversazioni chat-like testuali e trasferimento file.
La tecnologia di videoconferenza mette a disposizione i seguenti tipi di conversazione:

a) **Audio privato con ricezione automatica:**

   Rapporto con la risorsa: `isResource`

   Esclusività: `noLock`

   Numerosità: `multipleUtterance`

   Formato dati: `audio`

   Visibilità: `private`

   Aggiornamento: `realPush`

b) **Video privato con ricezione automatica:**

   Rapporto con la risorsa: `isResource`

   Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `video`

Visibilità: `private`

Aggiornamento: `realPush`

**c) Messaggi privati con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `private`

Aggiornamento: `realPush`

**d) Allegati privati con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `oggetto`

Visibilità: `private`

Aggiornamento: `realPush`

### 4.2.8 Editing condiviso

I collaboratori durante l'editing condividono un documento che assume stati diversi in seguito alle modifiche degli utenti. Un valore sarà quindi onResource, mentre il meccanismo di turni potrà essere localLock e multipleUtterance, o exclusiveLock e multipleUtterance.

Inoltre si consideri il caso in cui due collaboratori A e B stiano lavorando sullo stesso documento. A modifica il documento D, ottiene $D^1$ e lo invia a B.

B riceve il documento $D^1$, lo modifica e ottiene $D^2$. Contemporaneamente A continua a lavorare su $D^1$ ottenendo $D^3$.

Questa è senza dubbio una tipo di collaborazione in cui il rapporto con la risorsa è onResource e il meccanismo di turno è noLock.
Le conversazioni sono private, mentre l'aggiornamento potrebbe essere sia realPush che pull. Il primo è il caso in cui l'utente vuole essere avvisato ad ogni modifica, il secondo è il caso in cui l'utente non è interessato alla notifica immediata di modifiche ad altre parti del testo, e vuole richiederle manualmente quando necessarie.
Come spiegato precedentemente, alcune applicazioni permettono anche una ulteriore conversazione chat-like testuale per commenti o correzioni sul testo sotto modifica.
Dunque i tipi di conversazioni disponibili sono:

a) **Modifiche testuali locali private su una risorsa con ricezione automatica:**
   Rapporto con la risorsa: `onResource`
   Esclusività: `localLock`
   Numerosità: `multipleUtterance`
   Formato dati: `testo`
   Visibilità: `private`
   Aggiornamento: `realPush`

b) **Modifiche testuali locali private su una risorsa con ricezione su richiesta:**
   Rapporto con la risorsa: `onResource`
   Esclusività: `localLock`
   Numerosità: `multipleUtterance`
   Formato dati: `testo`
   Visibilità: `private`
   Aggiornamento: `pull`

c) **Modifiche testuali globali private su una risorsa con ricezione su richiesta:**
   Rapporto con la risorsa: `onResource`
   Esclusività: `globalLock`
   Numerosità: `multipleUtterance`
   Formato dati: `testo`

Visibilità: `private`

Aggiornamento: `pull`

**d) Modifiche testuali private su una risorsa con ricezione su richiesta:**

Rapporto con la risorsa: `onResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `private`

Aggiornamento: `pull`

**e) Modifiche testuali private su una risorsa con ricezione automatica:**

Rapporto con la risorsa: `onResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `private`

Aggiornamento: `realPush`

**f) Modifiche locali private su una risorsa con ricezione automatica:**

Rapporto con la risorsa: `onResource`

Esclusività: `localLock`

Numerosità: `multipleUtterance`

Formato dati: `oggetto`

Visibilità: `private`

Aggiornamento: `realPush`

**g) Modifiche locali private su una risorsa con ricezione su richiesta:**

Rapporto con la risorsa: `onResource`

Esclusività: `localLock`

Numerosità: `multipleUtterance`

Formato dati: `oggetto`

Visibilità: `private`

Aggiornamento: `pull`

**h) Modifiche globali private su una risorsa con ricezione su richiesta:**

Rapporto con la risorsa: `onResource`

Esclusività: `globalLock`

Numerosità: `multipleUtterance`

Formato dati: `oggetto`

Visibilità: `private`

Aggiornamento: `pull`

**i) Modifiche private su una risorsa con ricezione su richiesta:**

Rapporto con la risorsa: `onResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `oggetto`

Visibilità: `private`

Aggiornamento: `pull`

**j) Modifiche private su una risorsa con ricezione automatica:**

Rapporto con la risorsa: `onResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `oggetto`

Visibilità: `private`

Aggiornamento: `realPush`

**k) Comandi locali privati su una risorsa con ricezione automatica:**

Rapporto con la risorsa: `onResource`

Esclusività: `localLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `realPush`

**l) Comandi locali privati su una risorsa con ricezione su richiesta:**

Rapporto con la risorsa: `onResource`

Esclusività: `localLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `pull`

**m) Comandi globali privati su una risorsa con ricezione su richiesta:**

Rapporto con la risorsa: `onResource`

Esclusività: `globalLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `pull`

**n) Comandi privati su una risorsa con ricezione su richiesta:**

Rapporto con la risorsa: `onResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `pull`

**o) Comandi privati su una risorsa con ricezione automatica:**

Rapporto con la risorsa: `onResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `realPush`

**p) Messaggi privati con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `private`

Aggiornamento: `realPush`

### 4.2.9 Annotazioni

Come descritto precedentemente, le annotazioni sono commenti esterni alla risorsa a cui si riferiscono perciò il rapporto con la risorsa è aboutResource. I partecipanti possono liberamente inviare le proprie espressioni. E' possibile salvare le annotazioni sia su server pubblici in cui non ci sono restrizioni d'accesso, sia su server che consentono l'accesso attraverso login e password, quindi si può avere una conversazione privata o pubblica. L'aggiornamento è lasciato all'utente che sceglie di ottenere le nuove annotazioni quando ne ha bisogno.
Le conversazione fornite dalle annotazioni sono:

**a) Commenti privati con ricezione su richiesta:**

Rapporto con la risorsa: `aboutResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `private`

Aggiornamento: `pull`

**b) Commenti pubblici con ricezione su richiesta:**

Rapporto con la risorsa: `aboutResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `public`

Aggiornamento: `pull`

### 4.2.10 Whiteboard condivisa

La conversazione offerta dalla whiteboard non si riferisce ad una risorsa in particolare ma come nel caso delle chat o dell'instant messaging, si potrebbe salvare il risultato della collaborazione per creare una nuova risorsa. I partecipanti possono inviare tutte le espressioni che vogliono poiché non ci sono meccanismi di blocco e non c'è necessità di richiedere aggiornamenti perché questi sono inviati in tempo reale. In sintesi la conversazione che si può ottenere da una whiteboard è:

**a) Comandi privati con ricezione automatica:**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `realPush`

### 4.2.11 WikiWikiWeb

I wiki permettono due conversazioni che si differenziano per il valore del campo visibilità. Se il valore è public significa che il wiki è liberamente leggibile e

modificabile da tutti, altrimenti, se il valore è private significa che il wiki ha dei controlli d'accesso. Ogni pagina viene considerata come una risorsa che viene modificata durante la collaborazione. I wiki conservano la storia di ogni pagina in modo da poter facilmente ritornare ad una versione precedente. Quando un utente modifica una pagina questa viene bloccata per gli altri che possono continuare a vedere l'ultima versione salvata ma non possono fare modifiche finché la pagina non è disponibile. Questo è un lock globale e poiché l'utente con il turno può inviare quante espressioni vuole il valore dell'esclusività è multipleUtterance. Infine, il meccanismo di aggiornamento è pull perché ogni visitatore sceglie quando ricevere gli aggiornamenti richiedendo nuovamente la pagina al server.
Le conversazioni che sono messe a disposizione dai wiki sono:

a) **Modifiche testuali globali pubbliche su una risorsa con ricezione su richiesta:**
   Rapporto con la risorsa: `onResource`
   Esclusività: `globalLock`
   Numerosità: `multipleUtterance`
   Formato dati: `testo`
   Visibilità: `public`
   Aggiornamento: `pull`

b) **Allegati pubblici su una risorsa con ricezione su richiesta:**
   Rapporto con la risorsa: `onResource`
   Esclusività: `globalLock`
   Numerosità: `multipleUtterance`
   Formato dati: `oggetto`
   Visibilità: `public`
   Aggiornamento: `pull`

### 4.2.12 Calendari e schedulazione

La risorsa a cui ci si riferisce in questo tipo di tecnologia è il calendario e non ci sono lock perché tutti i partecipanti possono in qualunque momento inviare la propria disponibilità senza modifiche dirette alla risorsa solo riferendosi ad essa (*aboutResource*).
Non ci sono meccanismi di turno e la collaborazione è privata poiché solo chi è coinvolto in un particolare appuntamento ne è a conoscenza. Infine ogni partecipante sceglie liberamente quando ottenere gli aggiornamenti dunque il valore di questa dimensione è pull. Riassumendo, le conversazioni offerte sono:

**a) Allegati privati riguardanti una risorsa con ricezione su richiesta:**
Rapporto con la risorsa: `aboutResource`
Esclusività: `noLock`
Numerosità: `multipleUtterance`
Formato dati: `oggetto`
Visibilità: `private`
Aggiornamento: `pull`

**b) Allegati privati riguardanti una risorsa con ricezione su richiesta:**
Rapporto con la risorsa: `aboutResource`
Esclusività: `noLock`
Numerosità: `multipleUtterance`
Formato dati: `oggetto`
Visibilità: `private`
Aggiornamento: `realPush`

### 4.2.13 MMORPG

Nei MMORPG la risorsa che viene modificata è il mondo virtuale all'interno del quale si muovono i partecipanti; mondo costituito da vari oggetti con cui i personaggi interagiscono.
In alcuni casi i giocatori possono inviare contemporaneamente le loro espressioni poiché queste non interferiscono tra loro (si pensi a due giocatori che si trovano in posti diversi all'interno del gioco); in altri casi le azioni dei giocatori sono legate (ad esempio due giocatori che vogliono prendere lo stesso oggetto) e quindi regolate da blocchi ad alcune parti della risorsa. Avremo quindi sia conversazioni noLock che localLock.
La maggior parte di questi giochi fornisce inoltre la possibilità di inviare messaggi testuali che appaiono nella finestra di ogni partecipante. Le conversazioni sono:

a) **Comandi pubblici su una risorsa con ricezione automatica:**
   Rapporto con la risorsa: `onResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `comando`
   Visibilità: `public`
   Aggiornamento: `realPush`

b) **Comandi locali pubblici su una risorsa con ricezione automatica:**
   Rapporto con la risorsa: `onResource`
   Esclusività: `localLock`
   Numerosità: `multipleUtterance`
   Formato dati: `comando`
   Visibilità: `public`
   Aggiornamento: `realPush`

c) **Messaggi pubblici con ricezione automatica:**
   Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `public`

Aggiornamento: `realPush`

### 4.2.14 Giochi online

I giochi online forniscono un gran numero di conversazioni diverse che hanno in comune il rapporto con la risorsa (*isResource*) e il meccanismo di aggiornamento (*realPush*). Come nei MMORPG è possibile giocare e contemporaneamente mandare messaggi agli avversari sia privatamente che pubblicamente, quindi avremo delle conversazioni basate sui comandi e delle conversazioni testuali. Le conversazioni tramite comandi possono avere sia localLock che globalLock e le mosse possono essere sia pubbliche che private; il primo caso è, ad esempio, quello in cui altri utenti guardano la partita senza giocare. Le conversazioni che si possono avere con i giochi online sono:

**a) comandi singoli globali privati con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `globalLock`

Numerosità: `singleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `realPush`

**b) comandi singoli globali pubblici con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `globalLock`

Numerosità: `singleUtterance`

Formato dati: `comando`

Visibilità: `public`

Aggiornamento: `realPush`

**c) comandi globali privati con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `globalLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `realPush`

**d) comandi globali pubblici con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `globalLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `public`

Aggiornamento: `realPush`

**e) comandi singoli locali privati con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `localLock`

Numerosità: `singleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `realPush`

**f) comandi singoli locali pubblici con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `localLock`

Numerosità: `singleUtterance`

Formato dati: `comando`

Visibilità: `public`

Aggiornamento: `realPush`

**g) comandi locali privati con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `localLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `private`

Aggiornamento: `realPush`

**h) comandi locali pubblici con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `localLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `public`

Aggiornamento: `realPush`

**i) messaggi privati con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `private`

Aggiornamento: `realPush`

**j) messaggi pubblici con ricezione automatica**

Rapporto con la risorsa: `isResource`

Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `testo`

Visibilità: `public`

Aggiornamento: `realPush`

### 4.2.15 Desktop sharing

La collaborazione attraverso il desktop sharing non ha una risorsa a cui riferirsi perciò il rapporto è del terzo tipo (isResource). Non ci sono meccanismi di turno perché ogni partecipante può all'interno della propria finestra muovere il mouse liberamente, aprire programmi, creare, modificare o cancellare file e tutte le modifiche saranno riflesse nelle finestre degli altri partecipanti.
Il tipo di dato scambiato è comandi e la collaborazione è privata. Inoltre come detto le modifiche vengono inoltrate a tutti i partecipanti nello stesso momento in cui vengono effettuate, dunque il protocollo di update è realPush. In sintesi otteniamo che la conversazione messa a disposizione da questa tecnologia è:

a) **Comandi privati con ricezione automatica:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `comando`
   Visibilità: `private`
   Aggiornamento: `realPush`

### 4.2.16 Workflow system

I worflow system forniscono due tipi di conversazioni: da una parte permettono di gestire comandi che riguardano le risorse, ad esempio quando si specificano delle azioni che riguardano un processo; dall'altra permettono di collaborare organizzando il lavoro, quindi una conversazione isResource. La collaborazione può essere sia pubblica che

privata e il meccanismo di aggiornamento è realPush poiché i partecipanti vengono costantemente informati sui nuovi compiti da svolgere e sulle modifiche al sistema.
Le conversazioni fornite dai worflow system sono:

a) **Comandi pubblici riguardanti una risorsa con ricezione automatica:**
   Rapporto con la risorsa: `aboutResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `comando`
   Visibilità: `private`
   Aggiornamento: `realPush`

b) **Comandi privati riguardanti una risorsa con ricezione automatica:**
   Rapporto con la risorsa: `aboutResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `comando`
   Visibilità: `public`
   Aggiornamento: `realPush`

c) **Comandi pubblici con ricezione automatica:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`
   Numerosità: `multipleUtterance`
   Formato dati: `comando`
   Visibilità: `private`
   Aggiornamento: `realPush`

d) **Comandi privati con ricezione automatica:**
   Rapporto con la risorsa: `isResource`
   Esclusività: `noLock`

Numerosità: `multipleUtterance`

Formato dati: `comando`

Visibilità: `public`

Aggiornamento: `realPush`

## 4.3 Tabella riassuntiva

La tabella 1 mostra nella prima colonna le tecnologie collaborative analizzate e nella seconda colonna le conversazioni messe a disposizione da ciascuna di esse.

| Tecnologie | Conversazioni |
|---|---|
| Email | messaggi privati con ricezione su richiesta |
| | messaggi privati con ricezione su richiesta temporizzata |
| | allegati privati con ricezione su richiesta |
| | allegati privati con ricezione su richiesta temporizzata |
| Forum e Newsgroup | messaggi pubblici con ricezione su richiesta |
| | allegati pubblici con ricezione su richiesta |
| Blog | messaggi pubblici con ricezione su richiesta |
| | commenti pubblici con ricezione su richiesta |
| Chat | messaggi pubblici con ricezione automatica |
| | messaggi privati con ricezione automatica |
| | allegati pubblici con ricezione automatica |
| | allegati privati con ricezione automatica |
| | audio privato con ricezione automatica |
| | video privato con ricezione automatica |
| Instant Messaging | messaggi privati con ricezione automatica |
| | allegati privati con ricezione automatica |
| | audio privato con ricezione automatica |
| | video privato con ricezione automatica |
| Voice over IP | messaggi privati con ricezione automatica |
| | allegati privati con ricezione automatica |
| | audio privato con ricezione automatica |
| | video privato con ricezione automatica |
| Video Conferenza | audio privato con ricezione automatica |
| | video privato con ricezione automatica |

| | |
|---|---|
| Video Conferenza | messaggi privati con ricezione automatica |
| | allegati privati con ricezione automatica |
| Editing condiviso | modifiche testuali locali private su una risorsa con ricezione automatica |
| | modifiche testuali locali private su una risorsa con ricezione su richiesta |
| | modifiche testuali globali private su una risorsa con ricezione su richiesta |
| | modifiche testuali private su una risorsa con ricezione su richiesta |
| | modifiche testuali private su una risorsa con ricezione automatica |
| | comandi locali privati su una risorsa con ricezione automatica |
| | comandi locali privati su una risorsa con ricezione su richiesta |
| | comandi globali privati su una risorsa con ricezione su richiesta |
| | comandi privati su una risorsa con ricezione su richiesta |
| | comandi privati su una risorsa con ricezione automatica |
| | messaggi privati con ricezione automatica |
| Disegno condiviso | modifiche locali private su una risorsa con ricezione automatica |
| | modifiche locali private su una risorsa con ricezione su richiesta |
| | modifiche globali private su una risorsa con ricezione su richiesta |
| | modifiche private su una risorsa con ricezione su richiesta |
| | modifiche private su una risorsa con ricezione automatica |
| | comandi locali privati su una risorsa con ricezione automatica |
| | comandi locali privati su una risorsa con ricezione su richiesta |
| | comandi globali privati su una risorsa con ricezione su richiesta |
| | comandi privati su una risorsa con ricezione su richiesta |
| | comandi privati su una risorsa con ricezione automatica |
| Annotazioni | commenti pubblici con ricezione su richiesta |
| | commenti privati con ricezione su richiesta |
| Whiteboard | comandi privati con ricezione automatica |
| Wiki | modifiche testuali globali pubbliche su una risorsa con ricezione su richiesta |
| | allegati pubblici su una risorsa con ricezione su richiesta |

| | |
|---|---|
| Calendari | allegati privati riguardanti una risorsa con ricezione su richiesta |
| | allegati privati riguardanti una risorsa con ricezione automatica |
| MMORPG | comandi pubblici su una risorsa con ricezione automatica |
| | comandi locali pubblici su una risorsa con ricezione automatica |
| | messaggi pubblici con ricezione automatica |
| Giochi online | comandi singoli globali privati con ricezione automatica |
| | comandi singoli globali pubblici con ricezione automatica |
| | comandi globali privati con ricezione automatica |
| | comandi globali pubblici con ricezione automatica |
| | comandi singoli locali privati con ricezione automatica |
| | comandi singoli locali pubblici con ricezione automatica |
| | comandi locali privati con ricezione automatica |
| | comandi locali pubblici con ricezione automatica |
| | messaggi privati con ricezione automatica |
| | messaggi pubblici con ricezione automatica |
| Desktop Sharing | comandi privati con ricezione automatica |
| Workflow system | comandi pubblici riguardanti una risorsa con ricezione automatica |
| | comandi privati riguardanti una risorsa con ricezione automatica |
| | comandi pubblici con ricezione automatica |
| | comandi privati con ricezione automatica |

**Tabella 1 : Tecnologie e Conversazioni**

In questo capitolo è stato messo in evidenza come ogni tecnologia venga identificata univocamente da una serie di caratteristiche.

Nel prossimo capitolo si discuterà di come le parti che vogliono iniziare una conversazione possono comunicare le proprie capacità collaborative.

# Capitolo 5: I protocolli

In questo capitolo verrà presentata la descrizione dei protocolli che abbiamo creato per la nostra proposta, CAP e CUML, e verranno analizzate la struttura del profilo di collaborazione e la struttura delle espressioni che trasportano il contenuto.

## 5.1 Fasi della collaborazione

Quando due o più applicazioni vogliono iniziare una collaborazione, prima di scambiarsi i messaggi che la costituiscono, devono fornire le proprie capacità collaborative in modo che si possa verificare che le due parti abbiano funzionalità comuni tali da permettere la creazione di una nuova conversazione. L'interazione tra le parti, dunque, avviene in due fasi. Nella prima fase ogni parte redige il proprio profilo di collaborazione in cui specifica capacità e preferenze attraverso il protocollo CAP; nella seconda fase le parti inviano le espressioni in accordo con i profili usando il protocollo CUML.

Come i profili siano analizzati e i criteri con cui sia scelta una possibilità tra varie alternative sono lasciati all'implementazione.

Le parti spediscono il proprio profilo al server che risponde con il risultato del confronto. Il destinatario di una richiesta può o accettare la proposta o rifiutarla sia perché non ha caratteristiche comuni sia, nel caso di conversazioni tra più partecipanti, perché non vuole comunicare con gli altri invitati. Una volta che le parti si sono accordate si passa alla seconda fase, quella dello scambio di messaggi con contenuto.

Quello di cui ci occuperemo in questo capitolo è definire la struttura del profilo di collaborazione in relazione alle dimensioni descritte nel Capitolo 4, e la struttura delle espressioni scambiate tra le parti. A questo proposito sono stati creati due protocolli:

1. Il *protocollo di accordo collaborativo* (CAP, Collaboration Agreement Protocol) permette ai partecipanti di specificare il proprio profilo, cioè di definire il tipo di collaborazione che vogliono creare in accordo con i parametri di rapporto con la risorsa, meccanismo di turni, formato dati, meccanismo d'aggiornamento delle espressioni e visibilità;
2. Il *linguaggio collaborativo di markup per le espressioni* (CUML, Collaborative Utterance Markup Language) permette alle parti di contribuire con le espressioni coerenti con l'accordo di collaborazione distribuito in precedenza con successo alle parti.

Il funzionamento dei protocolli sarà spiegato in dettaglio nei prossimi paragrafi mentre la struttura e la descrizione degli elementi e degli attributi che costituiscono i documenti XML si potrà trovare nell'Appendice A.
Gli Xml Schema dei due protocolli saranno invece disponibili nell'Appendice B.
Dal paragrafo 5.4 in poi sarà analizzata nel dettaglio ogni fase della collaborazione.

## 5.2 Il protocollo di accordo collaborativo

Nel seguito ci riferiremo alla collaborazione come una o più conversazioni tra diversi partecipanti. Ogni conversazione è composta da un certo numero di espressioni, ognuna delle quali proviene da uno dei collaboratori.
Il numero di conversazioni durante una singola collaborazione può essere vario. Ad esempio una sofisticata applicazione di editing condiviso potrebbe mettere a disposizione una conversazione per i comandi sul testo più una conversazione chat-like tra gli autori. Ogni conversazione è totalmente indipendente dalle altre, e può avere caratteristiche completamente diverse.
Le parti si accordano sulle conversazioni impiegate nella collaborazione e per ognuna di esse:

- La relazione con la risorsa:
- L'URI della risorsa;
- Esclusività e numerosità nei turni;
- Il formato dati delle espressioni scambiate;
- Il protocollo di update delle espressioni;
- La granularità delle espressioni;
- La rinegoziabilità dell'accordo durante la collaborazione;
- Il protocollo di trasferimento usato.

Per comprendere meglio come venga creato il profilo di collaborazione di ogni partecipante analizziamo un esempio. Consideriamo un utente che possiede un programma di chat che permette lo scambio di messaggi testuali e l'invio di file.
Il nostro utente vuole comunicare con una sua amica ma non sa quali conversazioni ella possa effettuare. A questo proposito crea il suo profilo e lo invia al server che si occuperà di aspettare il profilo dell'altro partecipante, confrontarlo con quello del nostro primo utente ed inviare ad entrambi la risposta. I tipi di risposte possibili saranno analizzate meglio nel prossimo paragrafo.
Nella creazione del profilo si analizzano innanzitutto le conversazioni messe a disposizione dal software in uso. Nell'esempio la chat metterà a disposizione:

- messaggi privati con ricezione automatica
- allegati privati con ricezione automatica

Successivamente l'utente può specificare la preferenza di una conversazione o più rispetto alle altre presenti nel profilo attraverso l'attributo `favourite` o può scegliere il modo in cui essere aggiornato attraverso l'attributo `update`. Nel caso della chat generalmente il modello d'aggiornamento è `realPush` poiché le espressioni generate dall'altra parte vengono trasmesse non appena inviate ma l'utente può decidere di volerle ricevere ogni tot di tempo fissato o magari di richiederle manualmente. Inoltre l'utente potrebbe voler assicurare il destinatario sulla propria identità e a questo scopo inserire nel campo sender oltre al suo nome anche il suo certificato. Inoltre potrebbe voler garantire l'integrità e la confidenzialità dei dati che invia e perciò criptare e

firmare ogni messaggio. Il risultato della firma e della criptazione di un messaggio verranno mostrati nell'appendice C. Per il momento supponiamo che non siano necessari o desiderati meccanismi di sicurezza. Un esempio di profilo risultante potrebbe essere il seguente:

```
<cap:proposal xmlns:cap="http://www.cs.unibo.it/CAP/1.0/"
 renegotiable="true">
    <cap:sender uniqueName="Davide1978">
         <cap:name>Davide Rossi</cap:name>
    </cap:sender>
    <cap:request type="createNew"/>
    <cap:conversation
     id="a01"
     favourite="true"
     media="text/plain"
     quality="1.0">
         <cap:isResource>
             <cap:noLock>
                  <cap:multipleUtterance
                    update="realPush"
                    visibility="private"/>
                </cap:noLock>
        </cap:onResource>
        <cap:transport>
                <cap:transportProtocol version="1.1">HTTP
                </cap:transportProtocol>
                <cap:securityProtocol version="3.0">SSL
                </cap:securityProtocol>
        </cap:transport>
    </cap:conversation>
    <cap:conversation
     id="a02"
     favourite="true"
     media=" application/octet-stream"
     quality="1.0">
         <cap:isResource>
             <cap:noLock>
                 <cap:multipleUtterance
```

```
                update="realPush"
                visibility="private"/>
            </cap:noLock>
        </cap:isResource>
        <cap:transport>
            <cap:transportProtocol version="1.1">HTTP
            </cap:transportProtocol>
            <cap:securityProtocol version="3.0">SSL
            </cap:securityProtocol>
        </cap:transport>
    </cap:conversation>
</cap:proposal>
```

Ogni proposta di collaborazione può essere di due tipi. O il mittente vuole invitare un altro partecipante a creare una o più conversazioni oppure è già in corso una collaborazione e il mittente vuole invitare qualcun altro a partecipare. In questo caso il mittente vuole creare una nuova conversazione perciò il campo `Request` assume valore `createNew`.

Poiché la tecnologia permette due tipi di conversazione saranno presenti nel profilo due elementi `conversation`. Il primo elemento specifica l'invio di messaggi testuali. Questo si riconosce dal fatto che il valore dell'attributo `media` è `text/plain`. Le conversazioni che la chat può creare sono `isResource` infatti questi elementi sono presenti all'interno degli elementi `conversation`. Inoltre non vi sono meccanismi di turno, infatti le conversazioni sono `noLock`.

L'utente vuole ricevere i messaggi non appena disponibili perciò il valore del campo update è `realPush`. Inoltre si utilizza il protocollo HTTP per trasportare i messaggi e il protocollo SSL per garantire la confidenzialità.

A questo punto, dopo aver creato il proprio profilo, l'utente lo invia al server.

## 5.3 Confronto tra profili

Non appena il server riceve un profilo aspetta che il partecipante invitato invii il proprio, per poter, così, procedere al confronto.

Consideriamo come secondo partecipante un utente che possiede un programma di editing condiviso che permette oltre alla conversazione di modifiche sul testo anche una conversazione chat-like per lo scambio di messaggi testuali.
Maria Bianchi ha ricevuto una richiesta di comunicazione da Davide Rossi che aveva a disposizione un programma di chat per l'invio di messaggi e di oggetti. Per poter rispondere alla richiesta Maria deve creare il suo profilo e inviarlo al server che si occuperà di confrontare i due profili. Nella creazione del profilo anche in questo caso si analizzano innanzitutto le conversazioni messe a disposizione dal software in uso. Il programma di editing metterà a disposizione le seguenti conversazioni:

- modifiche testuali locali private su una risorsa con ricezione automatica
- messaggi privati con ricezione automatica

Anche Maria, a questo punto, specifica le sue preferenze per ogni conversazione che è capace di fare, indicando quali preferisce intraprendere e in che modo desidera essere aggiornata. Un esempio di profilo risultante potrebbe essere il seguente:

```
<cap:proposal xmlns:cap="http://www.cs.unibo.it/CAP/1.0/"
 renegotiable="true">
    <cap:sender uniqueName="Stellina">
         <cap:name>Maria Bianchi</cap:name>
    </cap:sender>
    <cap:request type="createNew"/>
    <cap:conversation
     id="b01"
     favourite="true"
     media=" application/x-text-editor"
     quality="1.0">
         <cap:onResource uri="http://www.sito.com/risorsa1">
             <cap:localLock grain="page" token="pool">
                 <cap:multipleUtterance
                  update="realPush"
                 visibility="private"/>
              </cap:localLock>
        </cap:onResource>
        <cap:transport>
```

```
            <cap:transportProtocol version="1.1">HTTP
            </cap:transportProtocol>
            <cap:securityProtocol version="3.0">SSL
            </cap:securityProtocol>
        </cap:transport>
    </cap:conversation>
    <cap:conversation
     id="b02"
     favourite="true"
     media="text/plain"
     quality="1.0">
         <cap:isResource>
             <cap:noLock>
                  <cap:multipleUtterance
                    update="realPush"
                    visibility="private"/>
                </cap:noLock>
        </cap:onResource>
        <cap:transport>
               <cap:transportProtocol version="1.1">HTTP
               </cap:transportProtocol>
               <cap:securityProtocol version="3.0">SSL
               </cap:securityProtocol>
        </cap:transport>
    </cap:conversation>

</cap:proposal>
```

Anche in questo caso sono presenti nel profilo due elementi `conversation` poiché la tecnologia permette due tipi di conversazione: quella di modifiche e quella come la chat. Quest'ultima ha gli stessi valori della conversazione testuale nel profilo del primo utente. La conversazione di modifiche sul testo, invece, è caratterizzata dai seguenti valori: la collaborazione è `onResource` e la risorsa su cui si vuole collaborare è identificata attraverso l'attributo `uri`. Per quanto riguarda i meccanismi di turno, le conversazioni sono `localLock` poiché si vuole che solo un utente alla volta possa modificare una determinata sezione ma si vuole anche lasciare i partecipanti liberi di effettuare modifiche contemporanee su parti diverse.

La dimensione delle sezioni in cui dividere il testo è stabilito nel campo `grain` e come si può osservare il valore è `page`. Questo significa che collaboratori potranno effettuare modifiche simultaneamente ma solo a patto di farlo in pagine differenti.
Ogni pagina sotto modifica verrà considerata come bloccata da un lock e tornerà disponibile non appena il partecipante con il turno lo rilascerà.
Il meccanismo dei token lascia che ogni utente decida quando modificare una parte libera; infatti, il valore di questo campo è `pool`.
L'utente vuole ricevere i messaggi non appena disponibili perciò il valore del campo update è, anche in questo caso, realPush e  si utilizzano i protocolli HTTP e SSL.

Non appena il secondo partecipante invia il suo profilo al server, si procede al confronto. In generale i profili potrebbero essere analizzati elemento per elemento partendo dall'inizio oppure si potrebbero seguire delle regole di precedenza. Una possibilità è quella di selezionare inizialmente solo le conversazioni preferite da ciascun utente. Queste ultime si differenziano da quelle che ogni partecipante è semplicemente capace di fare attraverso il valore `true` attribuito al campo `favourite`.
Nel caso non ci fosse corrispondenza tra le conversazioni preferite si potrebbe passare all'analisi delle restanti conversazioni disponibili. Come i due profili vengano effettivamente sottoposti al parsing, comunque, è un compito lasciato all'applicazione sul server e non ce ne occuperemo ulteriormente.

## 5.4 L'Agreement

L'analisi dei profili potrebbe concludersi con la generazione di un documento di accordo; altrimenti, l'analisi potrebbe concludersi con l'invio da parte del server di un responso alle parti senza la generazione di alcun documento di accordo.
Nel caso venga generato un documento, questo potrebbe risiedere sul server nel lasso di tempo necessario ad inviare alle parti il responso e poi cancellato subito dopo o potrebbe essere salvato come un documento vero e proprio ed essere inviato alle parti.
Ogni partecipante, a questo punto, dovrebbe analizzare il documento ricevuto per poter capire quali conversazioni potranno essere effettivamente utilizzate nella collaborazione. Anche quest'aspetto sarà lasciato all'implementazione dell'applicazione server.

A prescindere da come viene effettuata l'analisi dei profili e da come le parti ricevono le informazioni necessarie per poter iniziare la collaborazione, quello che si vuole mostrare è cosa si ottiene dal confronto.
Per questo scopo riprendiamo lo scenario descritto nei paragrafi precedenti in cui un utente aveva a disposizione una chat capace di inviare messaggi e allegati mentre l'altro utente aveva a disposizione un editor collaborativo con possibilità di fare modifiche sul documento condiviso e di inviare messaggi testuali chat-like.
L'unica conversazione che le parti hanno in comune è:

- messaggi privati con ricezione automatica

Questa conversazione sarà, dunque, quella che potrà essere intrapresa tre i due partecipanti.

## 5.5 Rifiuto

Alcune volte non può essere creata una conversazione e questo può dipendere da vari fattori. In primo luogo, banalmente, non è possibile intraprendere una collaborazione quando le parti non hanno nessuna caratteristica in comune.
Come detto prima è possibile che non ci siano conversazioni comuni tra quelle indicate come preferite da entrambe le parti ma potrebbero essere comunque disponibili conversazioni "di riserva". A questo punto la parte che si trova a poter instaurare solo una conversazione che non era tra le sue preferite può decidere di rifiutare l'invito.
Il modello che abbiamo creato permette di esprimere non solo il tipo di proposta che viene inviata, cioè se il mittente intende creare una nuova conversazione o vuole invitare il destinatario a partecipare ad una conversazione in corso, ma permette di specificare, nel primo caso, se altre persone sono state invitate e, nel secondo caso, chi sono le altre persone coinvolte nella conversazione in corso.
Grazie a questa possibilità il destinatario può scegliere di rifiutare l'invito nel caso non voglia conversare con uno degli invitati o con uno dei partecipanti della conversazione in corso.

Un altro motivo per cui non è possibile creare una collaborazione è che si sia verificato un problema con i certificati o con i meccanismi di sicurezza.

## 5.6 Il linguaggio collaborativo di markup delle espressioni

Una volta che le parti si sono accordate sulle conversazione che possono e vogliono intraprendere si passa alla collaborazione vera e propria in cui le parti si scambiano le espressioni che trasportano il contenuto. Ogni espressione, cioè ogni contributo individuale ad una conversazione, deve fornire le informazioni di contesto tutte le volte che viene inviata.
In particolare ogni conversazione deve fornire le seguenti informazioni:

- A quale conversazione appartiene;
- Chi la fornisce e quali certificati utilizza;
- La risorsa (se esiste) a cui si riferisce;
- I dati forniti.

Attraverso il linguaggio collaborativo di markup delle espressioni (CUML) è possibile fornire ai client un modo per esprimere queste informazioni.

## 5.7 Rinegoziazione dinamica dell'accordo

Una caratteristica del protocollo collaborativo è la capacità di rinegoziare l'accordo durante una conversazione. E' possibile in ogni momento modificare i parametri di collaborazione aumentando l'utilità, l'usabilità e la flessibilità delle applicazioni.
La rinegoziazione dinamica dell'accordo è esattamente la caratteristica che permette di associare tra loro applicazioni differenti e che rende possibile la trasformazione di un'applicazione nell'altra semplicemente modificando qualche regola di collaborazione.
Per fare degli esempi si considerino gli utenti di una chat sincrona. Essi potrebbero aggiungere una "cartolina" di disegno comune (che genera una nuova conversazione in aggiunta a quella di chat esistente), oppure potrebbero salvare la conversazione in corso trasformando la chat in un forum, o ancora potrebbero permettere ad altre persone di

partecipare alla conversazione, o modificare il meccanismo di turni o di blocco del documento.
Oppure si consideri un editor collaborativo sincrono che potrebbe essere trasformato in asincrono semplicemente cambiando qualche parametro.
L'obiettivo di questo progetto è di assicurare l'interoperabilità tra due o più utenti quando questi utilizzano tecnologie differenti o software di diversi produttori.

Per capire meglio l'utilità della rinegoziazione torniamo allo scenario presentatato nei paragrafi 5.2 e 5.3. Ricordiamo che il mittente ha a disposizione una chat che gli permette di inviare messaggi testuali e allegati mentre il destinatario della proposta usa un editor collaborativo che supporta anche conversazioni testuali chat-like. Dopo che i profili delle parti sono stati analizzati dal server e i collaboratori si sono accordati sono passati alla collaborazione vera e propria scambiandosi messaggi testuali in real-time senza meccanismi di turno. Supponiamo che a questo punto il mittente decida di prendere parte anche alla collaborazione sul testo utilizzando un editor comune che non permette di inviare messaggi chat.
Il fatto che il mittente utilizzi due applicazioni mentre il destinatario ne usi solo una che supporta entrambe le conversazioni non ha importanza ai fini della collaborazione.
La fase della rinegoziazione consiste nell'invio da parte del mittente di un nuovo profilo che questa volta è fatto come segue:

```
<cap:proposal xmlns:cap="http://www.cs.unibo.it/CAP/1.0/"
 renegotiable="true">
    <cap:sender uniqueName="Luca1982">
        <cap:name>Luca Rossi</cap:name>
    </cap:sender>
    <cap:request type="createNew"/>
    <cap:conversation
     id="a01"
     favourite="true"
     media="text/plain"
     quality="1.0">
         <cap:isResource>
             <cap:noLock>
                  <cap:multipleUtterance
```

```
                update="realPush"
                visibility="private"/>
            </cap:noLock>
    </cap:onResource>
    <cap:transport>
            <cap:transportProtocol version="1.1">HTTP
            </cap:transportProtocol>
            <cap:securityProtocol version="3.0">SSL
            </cap:securityProtocol>
    </cap:transport>
</cap:conversation>
<cap:conversation
 id="a02"
 favourite="true"
 media=" application/octet-stream"
 quality="1.0">
     <cap:isResource>
         <cap:noLock>
             <cap:multipleUtterance
              update="realPush"
              visibility="private"/>
          </cap:noLock>
    </cap:isResource>
    <cap:transport>
        <cap:transportProtocol version="1.1">HTTP
        </cap:transportProtocol>
        <cap:securityProtocol version="3.0">SSL
        </cap:securityProtocol>
    </cap:transport>
</cap:conversation>
<cap:conversation
 id="a03"
 favourite="true"
 media="application/x-text-editor"
 quality="1.0">
     <cap:onResource uri="http://www.sito.com/risorsa1">
         <cap:localLock grain="page" token="pool">
             <cap:multipleUtterance
              update="realPush"
```

```
                visibility="private"/>
            </cap:localLock>
        </cap:onResource>
        <cap:transport>
            <cap:transportProtocol version="1.1">HTTP
            </cap:transportProtocol>
            <cap:securityProtocol version="3.0">SSL
            </cap:securityProtocol>
        </cap:transport>
    </cap:conversation>

</cap:proposal>
```

Come si può notare, questa volta le conversazioni sono tre perché oltre alle due conversazioni offerte dalla chat ora l'utente ha anche a disposizione la conversazione offerta dall'editor.
Il profilo del destinatario è rimasto lo stesso ma a questo punto le parti condividono due conversazioni: quella testuale che era già in corso e la conversazione sul documento condiviso che si aggiunge alla prima.

# Capitolo 6: Conclusioni

Lo scopo di questa tesi è stato dimostrare come sia possibile collaborare utilizzando applicazioni differenti. Per dimostrare questa idea è stato creato un modello che permette di specificare le caratteristiche collaborative delle tecnologie e permette all'utente di esprimere le proprie preferenze.

Analizzando le varie tecnologie e le funzioni che queste mettono a disposizione si è notato come applicazioni di diversi produttori abbiano caratteristiche comuni. La classificazione delle funzionalità all'interno dello spazio della collaborazione è stato il primo passo da compiere. Attraverso la definizione delle dimensioni che costituiscono lo spazio si è potuto comprendere quali conversazioni fossero plausibili e quali invece fossero prive di senso.
Si è dimostrato che ogni conversazione può essere completamente descritta attribuendo dei valori alle dimensioni collaborative e dato un insieme di valori è possibile distinguere l'applicazione descritta.
Dopo aver identificato le dimensioni si è passati all'analisi individuale delle tecnologie in modo da poter evidenziare tutte le conversazioni possibili secondo l'applicazione utilizzata.

Questo è un modello preventivo che ammette ulteriori evoluzioni e implementazioni. Infatti, abbiamo notato che il protocollo di trasferimento utilizzato dalle parti ha una

grande importanza nel determinare il tipo e il successo di una applicazione collaborativa.

Le applicazioni sincrone generalmente implementano un protocollo di trasferimento bidirezionale orientato alla connessione in cui gli aggiornamenti non vengono richiesti ma inviati non appena disponibili. D'altra parte il Web si basa sul protocollo HTTP in cui le richieste sono inviate dal client attraverso un'architettura client-server.

La proposta fatta in questa tesi è indipendente dal protocollo di trasferimento ma le applicazioni dipendono fortemente da esso.

La scelta di usare SOAP per incapsulare i messaggi ha tenuto fortemente in considerazione questo aspetto. Il protocollo SOAP, infatti, anche se viene utilizzato soprattutto con i protocolli HTTP e SMTP, può essere usato con qualunque altro protocollo a livello sottostante (trasporto) e con altri protocolli a livello applicazione.

Affidarsi esclusivamente al protocollo HTTP potrebbe limitare la scalabilità dei protocolli proposti perché può essere usato in un numero limitato di attività collaborative relativamente alla sua intrinseca caratteristica di essere basato su una connessione iniziata dal client. Si è quindi capito che HTTP non è adeguato per applicazioni che necessitano aggiornamenti realPush e sarà necessario in futuro sviluppare un nuovo protocollo di trasferimento o modificare il protocollo HTTP. Questo protocollo dovrà poter essere usato al posto degli esistenti protocolli di trasferimento come HTTP, SMTP o NNTP.

L'aggiornamento pseudopush può risolvere al momento il problema di ottenere gli aggiornamenti non appena disponibili ma è molto dispendioso in termini di traffico sulla rete. Come detto in precedenza, l'aggiornamento pseudopush consiste nel far sì che l'applicazione richieda gli aggiornamenti ad ogni tot di tempo fissato dall'utente.

La dimensione del lasso di tempo ha una grande importanza: se è troppo lungo le applicazioni in ascolto potrebbero non ricevere in tempo espressioni inviate durante i tempi di attesa, se è troppo breve il protocollo implementa un meccanismo di busy-loop in cui gli aggiornamenti vengono richiesti in continuazione anche se nessun altro collaboratore ha prodotto nuove espressioni e quindi non ci sono dati da trasferire.

I busy-loop senza nessun nuovo dato da trasferire hanno chiaramente un grosso impatto sulle prestazioni della rete e non può essere esteso più di tanto.

L’aggiornamento pseudopush è una soluzione temporanea e sarà necessario introdurre un nuovo protocollo che supporti nativamente gli aggiornamenti realpush.
Il passo successivo sarà l’implementazione dei protocolli e dell’applicazione server che si occupa, come detto, del confronto tra i profili e dell’invio della risposta ai partecipanti.
L’applicazione server si deve anche occupare di fare delle scelte quando vi sono più valori disponibili per un singolo campo.

# Bibliografia


[AIM] AOL Instant messenger, http://www.aim.com/, 15 Novembre 2005.

[Ais02] Aissi S. et al. ”Collaboration-Protocol Profile and Agreement Specification Version 2.0”, 2002, http://www.oasis-open.org/committees/download.php/204/ebcpp-2.0.pdf, 2 Febbraio 2006.

[All01] Allen R. “Workflow: An introduction”, 2001, http://www.wfmc.org/information/Workflow-An_Introduction.pdf, 10 Febbraio 2006.

[Ann] Annotea. http://www.w3.org/2001/Annotea/, 12 Gennaio, 2005

[Ban93] Bannon L. “The Context of CSCW”, in: *Developing CSCW Systems: Design Concepts*, Schmidt K.(Ed), 1993, 9-36.

[BasShu04] Baset S.A., Shulzrinne H. “An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol”, 2004, http://arxiv.org/ftp/cs/papers/0412/0412017.pdf, 16 Febbraio 2006.

[BBFLS02] Bartel M., Boyer J., Fox B., LaMacchia B., Simon E. “XML-Signature Syntax and Processing”, W3C Recommendation, 12 Febbraio 2002, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/ 10 Febbraio 2006

[BHLT04] Bray T., Hollander D., Layman A., Tobin R. “Namespace in XML 1.1”, W3C Recommendation, 4 Febbraio 2004, http://www.w3.org/TR/xml-names11/, 1 Novembre 2005.

[Boy01] Boyer J. “Canonical XML Version 1.0”, W3C Recommendation, 15 Marzo 2001, http://www.w3.org/TR/2001/REC-xml-c14n-20010315 , 10 Febbraio 2006

[BriGom92] Brinck T., Gomez L. M., “The Design of the Conversation Board”, in: *Proceedings of Conference on Human Factors in Computing Systems*, Monterey, California, ACM, 3-5 Maggio 1992, 42.

[Bub02] Bub, A. S. “Computer Role-Playing games”, 2002,. http://www.gamespy.com/articles/april02/rpgweek/rpg1/ , 25 Settembre 2005.

[CDP04] Curran K., Doherty K., Power R. “WikiWikiWeb as a Tool for Collaboration”, *Information Technology Journal*, 3(2), 2004, 206-210.

[Cri04] Crispin M. “Internet Message Access Protocol - Version 4rev1”, RFC3501, IETF, 2004, http://www.faqs.org/rfcs/rfc3501.html, 20 Dicembre 2005.

[CRSHG02] Campbell B., Rosenberg J., Schulzrinne H., Huitema C., Gurle D. “Session Initiation Protocol (SIP) Extension for Instant Messaging”, RFC3428, IETF, 2002, http://www.ietf.org/rfc/rfc3428.txt, 12 Dicembre 2005.

[Cuc02] Cuciz, D. “The History of MUDs”, 2002, http://www.gamespy.com/articles/january01/muds1/ , 25 Settembre 2005.

[DBBO96] Dillenbourg P., Baker M., Blaye A., O'Malley C. “The evolution of research on collaborative learning”, in: *Learning in Humans and Machine: Towards an interdisciplinary learning science*, E. Spada & P. Reiman (Eds), Oxford, Elsevier, 1996, 189-211.

[DWF03] Dewes C., Wichmann A., Feldmann A. “An Analysis of Internet chat system” in: *Proceedings of Internet Measurement Conference* , Miami, Florida, ACM, 27-29 Ottobre 2003, 51-64.

[EGR91] Ellis C. A., Gibbs S.J., and Rein G.L. “Groupware: Some Issues and Experiences”, *Communications of ACM*, 34(1), 1991, 40.

[FGMFB97] Fielding R., Gettys J., Mogul J., Frystyk H., Berners-Lee T. “Hypertext Transfer Protocol -- HTTP/1.1”, RFC 2068, IETF, 1997, http://rfc.net/rfc2068.html, 23 Febbraio 2006.

[FreBor96] Freed N., Borenstein N. ” Multipurpose Internet Mail Extensions (MIME) Part Two:Media Types”, RFC2046, IETF, 1996, http://www.faqs.org/rfcs/rfc2046.html, 17 Febbraio 2006.

[GAI] Gaim, http://gaim.sourceforge.net/about.php, 15 Novembre 2005.

[GHMMN03a] Gudgin M., Hadley M., Mendelsohn N., Moreau J., Nielsen H. F. “SOAP Version 1.2 Part 1: Messaging Framework”, W3C Recommendation, 24 Giugno 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, 6 Febbraio 2006

[GHMMN03b] Gudgin M., Hadley M., Mendelsohn N., Moreau J., Nielsen H. F. “SOAP Version 1.2 Part 2: Adjuncts”, W3C Recommendation, 24 Giugno 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624/, 6 Febbraio 2006

[Gim03] Gimson R. “Device Independence Principles”, W3C Working Group Note, 1 Settembre 2003, http://www.w3.org/TR/2003/NOTE-di-princ-20030901/, 12 Gennaio 2006.

[Gla]Glance, http://www.glance.net/, 12 Febbraio 2006.

[Gra92] Graham R. “Collage: Foundational metapplication”, 1992, http://www.ncsa.uiuc.edu/News/Access/Archive/backissues/92.4/92.4Collage.html, 12 Febbraio 2006.

[Gri00] Griffiths R. “Computer Supported Co-operative Work and Groupware”, 2000, http://www.it.bton.ac.uk/staff/rng/teaching/notes/CSCWgroupware.html, 10 Febbraio 2006.

[GriPal02] Grinter R., Palen L. “Instant Messaging in Teen Life” in: *Proceedings of Conference on Computer Supported Collaborative Work,* New Orleans, Louisiana, ACM, 16-20 Novembre 2002, 21-30.

[HolMut98] Holtman K., Mutz A. “Trasparent Content Negotiation in HTTP”, RFC 2295, IETF, 1998, http://rfc.net/rfc2295.html, 23 Febbraio 2006

[HSBW04] Herring S., Scheidt L., Bonus S., Wright E. “Bridging the gap: A genre analysis of weblogs” in: *Proceedings of Hawaii International Conference on System Sciences,* Big Island, Hawaii, IEEE Computer Society, 5-8 Gennaio 2004, 1-11.

[ICQ] ICQ, http://www.icq.com/, 15 Novembre 2005.

[IDS02] Imamura T., Dillaway B., Simon E. “XML Encryption Syntax and Processing”, W3C Recommendation, 10 Dicembre 2002, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/ , 10 Febbraio 2006

[IWWSK02] Isaacs E., Walendowski A., Whittaker S., Schiano D.J., Kamm C. “The Character, Functions, and Styles of Instant Messaging in the Workplace” in: *Proceedings of Conference on Computer Supported Collaborative Work,* New Orleans, Louisiana, ACM, 16-20 Novembre 2002, 11-20.

[Jab] Jabber, http://www.jabber.org/, 22 Gennaio 2006.

[KanLap86]Kantor B., Lapsley P. “Network News Transfer Protocol” RFC977, IETF, 1986, http://www.faqs.org/rfcs/rfc977.html, 20 Dicembre 2005.

[KanKoi01] Kahan J., Koivunen M.R. “Annotea: An Open RDF Infrastucture for Shared Web Annotations” in *Proceedings of the International World Wide Web Conference*, Hong Kong, ACM, 1-5 Maggio 2001, 623-632.

[Kle01] Klensin J., ” Simple Mail Transfer Protocol”, RFC2821, IETF, 2001, http://www.ietf.org/rfc/rfc2821.txt, 20 Dicembre 2005.

[Kuh02] Kuhlmann R. ” ICQ/OSCAR protocol”, 2002, http://www.micq.org/ICQ-OSCAR-Protocol-v7-v8-v9/, 21 Ottobre 2005.

[KOP] Kopete, http://kopete.kde.org/, 15 Novembre 2005.

[KRWOHBT04] Klyne G., Reynolds F., Woodrow C., Ohto H., Hjelm J., Butler M. H., Tran L. ”Composite Capability/Preference Profiles: Structure and Vocabularies 1.0” W3C Recommendation, 15 Gennaio 2004, http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/, 11 Febbraio 2006.

[Lam04] Lamb B. “Wide open spaces wikis ready or not”, *Educause Review,* 39(5), 2004, 36-48.

[Mat05] Mattila E. “An Analysis of Hybrid and Pure Peer-to-Peer Technologies for IP Telephony”, 2005, http://www.tml.tkk.fi/Publications/C/18/mattila.pdf, 27 Ottobre 2005

[Mit03] Mitra N. “SOAP Version 1.2 Part 0: Primer” W3C Recommendation, 24 Giugno 2003, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/ , 6 Febbraio 2006.
[MyeRos96]Myers J., Rose M. “Post Office Protocol - Version 3”, RFC1939, IETF, 1996, http://www.faqs.org/rfcs/rfc1939.html, 20 Dicembre 2005.

[OikRee93] Oikarinen J., Reed D., “IRC: Internet Relay Chat Protocol” RFC1495, IETF, 1993, http://www.ietf.org/rfc/rfc1495.txt?number=1495 , 25 Ottobre 2005.

[Pra99] Prakash A. “Group Editors” in: *Computer Supported Cooperative Work*, M. Beaudouin-Lafon (Ed), New York, John Wiley & Sons Ltd, 1999, 103-133.

[RKC02] Rose M., Klyne G., Crocker D. “The Application Exchange Core”, RFC3340, IETF, 2002, http://www.ietf.org/rfc/rfc3340.txt, 15 Dicembre 2005.

[RSCJPSH02] Rosemberg J., Schulzrinne H., Camarillo G., Johnston A., Peterson J., Sparks R., Handley M. “SIP: Session Initiation Protocol” RFC3261, IETF, 2002, http://www.ietf.org/rfc/rfc3261.txt?number=3261 , 2 Gennaio 2006.

[Sai04] Saint-Andre P. “Extensible Messaging and Presence Protocol (XMPP): Core”, RFC3920, IETF, 2004, http://www.ietf.org/rfc/rfc3920.txt, 14 Gennaio 2006.

[SHK04] Schroeter R., Hunter J., Kosovic D. “FilmEd – Collaborative Video Indexing, Annotation and Discussion Tools Over Broadband Networks” in *Proceedings of International Multimedia Modelling Conference*, Brisbane, Australia, IEEE Computer Society, 5-7 Gennaio 2004, 346 – 353.

[Show99] http://www.agocg.ac.uk/reports/mmedia/video5/chap2a.htm, 1999, 12 Febbraio 2006.

[Skype] Skype. http://www.skype.com, 2 Gennaio 2006.

[TRI] Trillian, http://www.ceruleanstudios.com/, 15 Novembre 2005.

[VNC] RealVNC, http://www.realvnc.com/, 12 Febbraio 2006.

[WB02] http://www.freshports.org/mbone/wb/, 2002,12 Febbraio 2006.

[WenGen04] Weng C., Gennari J.H. “Asynchronous Collaborative Writing through Annotations” in: *Proceedings of conference on Computer Supported Cooperative Work* Chicago, Illinois, ACM, 6-10 Novembre 2004, 578 – 581.

[Wik] Wikipedia. http://www.wikipedia.org, 15 Novembre 2005.

[YIM] Yahoo! Messenger, http://it.messenger.yahoo.com/, 15 Novembre 2005.

# Appendice A: Struttura dei protocolli

Nei prossimi paragrafi verranno descritti dettagliatamente il CAP e il CUML e verranno analizzati gli elementi e gli attributi che li compongono. Le proposte e le utterance sono documenti XML incapsulati in messaggi SOAP.

## 1 Namespace

La dichiarazione dei Namespace[BHLT04] è obbligatoria.
Per il documento CAP è:
`xmlns:cap="http://www.cs.unibo.it/CAP/1.0/"`
Per il documento CUML è:
`xmlns:cuml="http://www.cs.unibo.it/CUML/1.0/"`

## 2 CAP

In questo protocollo si definisce il formato della proposta di collaborazione. Attraverso il CAP ogni partecipante descrive le proprie capacità collaborative e le proprie preferenze. Nel CAP tutte le informazioni relative al profilo sono contenute nella sezione header del pacchetto SOAP mentre il body è vuoto.

### *1.1 Struttura generale*

Segue la struttura del CAP. Se non specificato diversamente gli elementi del CAP devono essere presenti nell'ordine mostrato. Nei prossimi paragrafi descriveremo ogni elemento con maggior dettaglio.

```
<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
  http://schemas.xmlsoap.org/soap/envelope/"
```

```
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Header >
      <cap:proposal xmlns:cap="http://www.cs.unibo.it/CAP/1.0/"
        renegotiable="...">
          <cap:sender uniqueName="..."> </cap:sender>
          <cap:request type"..."> </cap:request>
          <cap:conversation
               id="..."
               favourite="..."
               media="..."
               quality=”...”
               combineWith="...">
          </cap:conversation>   <!-- una o più -->
          <cap:security> </cap:security>
      </cap:proposal>
    </soap:Header>
   <soap:Body/>
</soap:Envelope>
```

### *1.2 Elemento proposal*

L’elemento **proposal** è la radice del documento XML CAP.

In aggiunta l’elemento **proposal** contiene un attributo obbligatorio **renegotiable** che può assumere valore “**true**” oppure “**false**”. Se il valore è **true** significa che la proposta è rinegoziabile, dunque che è possibile durante una conversazione inviare una nuova proposta per modificare qualche parametro di collaborazione. Se il valore è **false** non sarà possibile rinegoziare l’accordo.
L’elemento **proposal** è composto dai seguenti elementi:

- Un elemento obbligatorio **sender** che identifica il mittente della proposta;
- Un elemento obbligatorio **request** che specifica il tipo di proposta inviata;
- Uno o più elementi **conversation** di cui uno obbligatorio che identificano le conversioni che ogni parte vuole ed è capace di effettuare;

- Un elemento opzionale **security** all'interno del quale vengono specificati i parametri di sicurezza.

Un documento CAP può essere firmato digitalmente fornendo i mezzi per accertare l'integrità, cioè per assicurare che il documento non sia stato modificato, e per accertare l'identità dell'autore. Un CAP può essere firmato utilizzando tecnologie conformi alla specifica XML Digital Signature[BBFLS02].
Un documento CAP può anche essere criptato per assicurare confidenzialità ai dati scambiati. Per la criptazione si utilizzerà la specifica XML Encryption[IDS02]

### *1.3 Elemento sender*

L'elemento **sender** identifica il mittente della proposta le cui capacità sono descritte in questo CAP. Ci può essere solo un elemento **sender** in ogni CAP ed è obbligatorio.
La struttura dell'elemento **sender** è la seguente:

```
<cap:sender uniqueName="...">
        <cap:name>…</cap:name>
        <cap:certificate>
             <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
              ...
             </ds:KeyInfo>
       </cap:certificate>
</cap:sender>
```

L'elemento **sender** contiene un attributo obbligatorio **uniqueName** che identifica univocamente ogni partecipante ed è formato dai seguenti sotto elementi:

- Un elemento **name** che indica il nome comune del partecipante. A differenza dell'**uniqueName** il **name** può non essere unico.
- Un elemento opzionale **certificate** che contiene il certificato del sender secondo le specifiche XML Digital Signature

### *1.4 Elemento request*

Ogni proposta di collaborazione può essere di due tipi. O il mittente vuole invitare un altro partecipante a creare una o più conversazioni oppure è già in corso una collaborazione e il mittente vuole invitare qualcun altro a partecipare.
Per specificare queste due possibilità è stato introdotto l'elemento request che ha la seguente struttura:

```
<cap:request type="...">
        <cap:invited>
                <cap:party uniqueName="..."
                        <cap:name>...</cap:name>
                        <cap:certificate>
                                <ds:KeyInfo
                        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                                 ...
                                </ds:KeyInfo>
                        </cap:certificate>
                </cap:party>
                <cap:party uniqueName="...">
                        <cap:name>...</cap:name>
                        <cap:certificate>
                                <ds:KeyInfo
                        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                                ...
                                </ds:KeyInfo>
                        </cap:certificate>
                </cap:party>
        </cap:invited>
</cap:request>
```

**Request** ha un attributo obbligatorio **type** che può assumere i valori "**createNew**" o "**participate**". Se il valore è **createNew** significa che il mittente propone di creare una o più nuove conversazioni con le caratteristiche specificate negli elementi **conversation**. Nel caso il valore sia "**participate**", il mittente sta invitando il destinatario a partecipare ad una o più conversazioni già in atto tra il mittente e uno o più altri partecipanti.

All'interno dell'elemento **request** vi è un elemento obbligatorio invited che assume diversi significati a seconda del valore di **type**. Se **type** è **createNew** allora l'elemento **invited** contiene tanti **party** quante sono le persone che sono state invitate dal mittente oltre al destinatario della proposta. Ogni invitato riceve una copia della proposta identica in tutti i campi eccetto l'elemento **invited**, quindi il solo scopo di questo elemento è quello di informare il destinatario della possibilità che altri partecipanti accettino la proposta e partecipino alla collaborazione. Se **type** è, invece, **participate** significa che è già in atto una conversazione tra il mittente e uno o più altri partecipanti elencati all'interno dell'elemento **invited**. Poiché per essere **participate** si suppone che ci sia almeno un terzo partecipante, oltre a mittente e destinatario, coinvolto nella collaborazione è obbligatoria la presenza di almeno un **party** all'interno dell'elemento **invited**. Gli elementi **party** hanno la stessa struttura dell'elemento **sender** precedentemente descritto. Riassumendo, se la proposta è di tipo **createNew** l'elemento **invited** può essere vuoto; se la proposta è di tipo **participate** allora deve essere presente almeno un **party** all'interno di **invited**.

### *1.5 Elemento conversation*

I partecipanti possono voler creare diverse conversazioni all'interno della stessa collaborazione. Ad esempio si potrebbe voler creare una conversazione testuale real-time più una conversazione con scambio di immagini. Per ogni conversazione che si intende proporre si inserisce un elemento **conversation** con la seguente struttura:

```
<cap:conversation
   id="..."
   favorite="..."
   media="..."
   quality="..."
   combineWith="... ... ...">

        <cap:onResource uri="..." >
               <cap:localLock grain="..." token="...">
                      <cap:multipleUtterance
                         update="..."
                         visibility="..."/>
```

```
                </cap:localLock>
        </cap:onResource>
        <cap:transport>
                <cap:transportProtocol version="..."> ...
                </cap:transportProtocol>              <!--Uno o più-->
                <cap:securityProtocol version="..."> ...
                </cap:securityProtocol>
        </cap:transport>
</cap:conversation>
```

**Conversation** contiene i seguenti attributi tutti obbligatori a parte **quality** e **combineWith** che sono opzionali:

- **Id**: identificativo univoco per ogni conversazione;
- **Favorite**: tramite questo campo è possibile specificare le conversazioni preferite tra quelle disponibili. Può assumere valore "**true**" oppure "**false**";
- **Media**: descrive il tipo di dato secondo la specifica MIME[FreBor96];
- **Quality:** specifica la qualità del formato. Si consideri, per fare un esempio, una conversazione basata sullo scambio di immagini JPEG: se un partecipante è in grado di supportare solo immagini in un formato differente queste potrebbero essere trasformate dall'applicazione con una qualità che dipende dal formato. I valori possibili sono **1.00**, **0.80**, **0.50**, **0.30**, **0.00**, dove 1.00 indica un'ottima qualità e 0.00 indica una qualità pessima;
- **combineWith**: attraverso questo attributo si possono raggruppare più conversazioni che possono o devono essere eseguite contemporaneamente, ad esempio una conversazione audio insieme alla corrispondente conversazione video. Il formato di questo campo è lo stesso del campo **id** e, in particolare, la conversazione inserita nel campo **combineWith** deve essere presente tra le **conversation** all'interno dello stesso CPA e tra le due o più deve esserci una corrispondenza biunivoca. In pratica, se la conversazione 1 ha la conversazione 2 specificata come **combineWith** allora la conversazione 2 avrà la conversazione 1 nel suo **combineWith**.

All'interno dell'elemento **conversation** troviamo:

- Uno tra gli elementi **onResource**, **isResource**, **aboutResource**. Almeno uno di questi elementi deve apparire obbligatoriamente, e non ne può apparire più di uno.
- Un elemento obbligatorio **transport** che specifica il tipo di protocolli usati per il trasferimento.

### *1.6 Elemento onResouce*

Come descritto nel paragrafo 4.1.1 il rapporto con la risorsa specifica come i collaboratori interagiscono durante la collaborazione.
Per descrivere i diversi comportamenti possibili sono stati introdotti tre elementi **onResource**, **isResource** e **aboutResource**. Questi tre elementi si somigliano nella struttura ma secondo il tipo di rapporto variano gli attributi e i sottoelementi.
La sintassi dell'elemento **onResource** è la seguente:

```
<cap:onResource uri="..." >
        <cap:localLock grain="..." token="...">
                <cap:multipleUtterance
                    update="..."
                     visibility="..."/>
        </cap:localLock>
</cap:onResource>
```

L'elemento **onResource** identifica conversazioni che avvengono su una risorsa identificata attraverso l'attributo obbligatorio **uri**.
Questo elemento è composto di un unico sottoelemento a scelta tra:

- Un elemento **noLock** che significa che non ci sono meccanismi di turno sulla risorsa.
- Un elemento **localLock** che esprime la possibilità di bloccare alcune parti della risorsa.
- Un elemento **globalLock** che indica la possibilità di bloccare l'intera risorsa.

### *1.7 Elemento isResource*

L'elemento **isResource** identifica collaborazioni che non hanno una risorsa vera e propria a cui riferirsi, ma il risultato delle conversazione può, volendo, essere salvato per creare una risorsa.
La struttura di **isResource** è la seguente:

```
<cap:isResource>
        <cap:noLock>
                <cap:multipleUtterance update="..." visibility="..."/>
         </cap:noLock>
</cap:isResource>
```

L'elemento **isResource** non contiene  attributi e contiene un sottoelemento a scelta tra **noLock**, **localLock** e **globalLock**, come nel caso di **onResource**.

### *1.8 Elemento aboutResource*

Le conversazioni **aboutResource** si svolgono riferendosi ad una risorsa che non viene modificata durante la collaborazione. La struttura è:

```
<cap:aboutResource uri="..." >
        <cap:noLock>
                <cap:multipleUtterance update="..." visibility="..."/>
        </cap:noLock>
</cap:aboutResource>
```

L'elemento **aboutResource** ha un attributo obbligatorio uri che identifica la risorsa a cui ci si riferisce. L'unico elemento figlio di **aboutResource** è **noLock**. Questo dipende dal fatto che qualunque conversazione che fa riferimento ad una risorsa permette per sua natura il libero invio di espressioni, poiché queste, in qualunque numero, di qualunque tipo e provenienti da qualunque partecipante non si influenzano tra loro né modificano la risorsa a cui si riferiscono.

### *1.9 Elemento noLock*

L'elemento **noLock** identifica conversazioni sincrone in cui tutti i partecipanti possono liberamente inviare le proprie espressioni.

```
<cap:noLock>
        <cap:multipleUtterance update="..." visibility="..."/>
</cap:noLock>
```

L'unico figlio dell'elemento **noLock** è **multipleUtterance** poiché non essendoci meccanismi di turno ogni partecipante può produrre quante espressioni desidera.

### *1.10 Elemento localLock*

L'elemento **localLock** specifica che è possibile produrre espressioni ma queste sono: limitate a parti di una risorsa nel caso di collaborazione **onResource**; oppure limitate alle parti della risorsa temporanea nel caso di collaborazione **isResource**.
La sintassi dell'elemento è la seguente:

```
<cap:localLock grain="..." token="...">
        <cap:multipleUtterance update="..." visibility="..."/>
</cap:localLock>
```

Questo elemento ha due attributi obbligatori:

- **Grain**: permette di specificare la granularità delle sezioni in cui viene suddivisa una risorsa (vera e propria o provvisoria). Il tipo di grain dipende dal formato dati poiché per ogni formato ci possono essere diversi tipi di grain. Ad esempio se il formato dati è testo allora il grain potrebbe essere sul carattere, sulla parola, sulla pagina o sul capitolo.
- **Token**: Nel caso in cui ci sia un meccanismo di turno, il testimone passa tra un partecipante e l'altro in diversi modi. In alcuni casi potrebbe essere il partecipante con il turno a scegliere a chi passarlo o in altri casi questo meccanismo potrebbe essere trasparente agli utenti. Abbiamo riscontrato quattro casi possibili, che sono anche i valori che può assumere l'attributo grain:

- **Pool:** ogni volta che un partecipante con il turno finisce di inviare le sue espressioni rilascia il turno e tutti gli altri se lo desiderano hanno la possibilità di prenderlo. Poiché potrebbero volerlo contemporaneamente in due o più solo il primo che richiede il turno riesce ad ottenerlo.
- **Client:** In generale si può immaginare che ogni qualvolta il turno viene rilasciato venga inviato un messaggio "tocca a te". In questo caso è l'applicazione client del partecipante con il turno che sceglie a chi inviare il messaggio.
- **User:** è l'utente che scegli a chi passare il turno;
- **Server:** In questo caso il messaggio "tocca a te" viene inviato dall'applicazione server in modo trasparente agli utenti. E' il caso, ad esempio, dei giochi di carte in cui il turno passa sempre o al giocatore a destra o al giocatore a sinistra.

All'interno di localLock troviamo a scelta:

- Un elemento **singleUtterance** che indica che è possibile produrre una sola espressione;
- Un elemento **multipleUtterance** che indica che possono essere prodotte più espressioni.

### *1.11 Elemento globalLock*

L'elemento **globalLock** indica che un solo partecipante per turno ha la possibilità di produrre espressioni.

```
<cap:globalLock token="...">
        <cap:singleUtterance update="..." visibility="..."/>
</cap:globalLock>
```

**GlobalLock** ha un attributo obbligatorio **token** descritto nel paragrafo precedente e a scelta uno tra gli elementi **singleUtterance** e **multipleUtterance**.

### *1.12 Elementi singleUtterance e multipleUtterance*

Questi elementi hanno content model empty e specificano esclusivamente se il partecipante con il turno può inviare una singola espressione o può inviarne diverse.

```
<cap:singleUtterance update="..." visibility="..."/>
<cap:multipleUtterance update="..." visibility="..."/>
```

Ognuno di questi elementi ha due attributi obbligatori:

- **Update**: indica il modello di aggiornamento come descritto nel paragrafo 3.1.5. e può dunque assumere uno tra i seguenti valori:
    - **Pull**: gli aggiornamenti vengono richiesti manualmente dall'utente;
    - **PseudoPush:** gli aggiornamenti sono richiesti dall'applicazione client ad ogni tot di tempo specificato dall'utente;
    - **RealPush:** gli aggiornamenti vengono inviati non appena disponibili.

- **Visibility**: descrive la visibilità delle espressioni. Può assumere valore "public" o "private" con il significato spigato nel paragrafo 4.1.4.

### *1.13 Elemento security*

L'elemento security permette di specificare i meccanismi utilizzati per garantire confidenzialità, autenticazione e integrità dei documenti XML inviati durante la collaborazione.
Questo elemento è opzionale poiché le parti potrebbero collaborare all'interno di un canale sicuro, quindi non avere bisogno di ulteriori meccanismi di sicurezza.
La struttura dell'elemento security è la seguente:

```
<cap:security>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        ...
      </ds:Signature>
```

```
      <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
        ...
      </enc:EncryptedKey>
</cap:security>
```

All'interno dell'elemento security possiamo trovare:

- Un elemento ds:Signature con namespace obbligatorio http://www.w3.org/2000/09/xmldsig# secondo la specifica XML Digital Signature che permette di firmare il documento;
- Un elemento enc:EncryptedKey con namespace obbligatorio http://www.w3.org/2001/04/xmlenc# secondo la specifica XML Encryption che permette di criptare il documento;
- Entrambi gli elementi appena citati che permettono di firmare e criptare il documento.

# 3 CUML

In questo protocollo si definisce il formato delle espressioni che vengono scambiate durante la collaborazione dopo che le parti si sono accordate sulle loro capacità e preferenze. Attraverso il CUML ogni partecipante ha una struttura standard per le espressioni con contenuto. Nel CUML tutte le informazioni relative alla collaborazione sono contenute nella sezione header del pacchetto SOAP mentre il body contiene il contenuto vero e proprio.

## *3.1 Struttura generale*

La struttura generale di un documento CUML è la seguente:

```
<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
   http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

  <soap:Header>
     <cuml:utterance
     xmlns:cuml="http://www.cs.unibo.it/CUML/1.0/">
```

```
              <cuml:conversation id="..."/>
              <cuml:sender uniqueName="..."/>
              <cuml:resource uri="..."/>
              <cuml:content media="..."/>
    </cuml:utterance>
  </soap:Header>

  <soap:Body>
        <html>
          <head><title>esempio</title></head>
          <body><p>Hello world</p></body>
        </html>
  </soap:Body>
</soap:Envelope>
```

### *3.2 Elemento utterance*

L'elemento **utterance** è la radice del documento XML CUML.

La struttura dell'elemento **utterance** è la seguente:

```
    <cuml:utterance
    xmlns:cuml="http://www.cs.unibo.it/CUML/1.0/">

              <cuml:conversation id="..."/>
              <cuml:sender uniqueName="..."/>
              <cuml:resource uri="..."/>
              <cuml:content media="..."/>
              <cuml:security/>
    </cuml:utterance>
```

L'elemento **utterance** è composto dai seguenti elementi:

- Un elemento obbligatorio **conversation**: indica la conversazione alla quale il messaggio si riferisce;
- Un elemento obbligatorio **sender**: identifica il mittente del messaggio;
- Un elemento opzionale **resource**: se esiste identifica la risorsa alla quale si riferisce il messaggio;
- Un elemento opzionale **content**: indica il formato dei dati contenuti nel body del messaggio;
- Un elemento opzionale **security**: ha la stessa struttura e la stessa funzione dell'elemento security che si trova nel CAP.

### *3.3 Elemento conversation*

Questo elemento obbligatorio ha lo scopo di identificare la conversazione a cui si riferisce l'espressione contenuta nel body del pacchetto SOAP.
Ogni espressione è associata ad un identificatore univoco e il suo valore deve essere uguale ad uno degli **id** di conversazione proposti nel CAP.
L'elemento conversation non ha sottoelementi ed ha un unico attributo **id**.

### *3.4 Elemento sender*

L'elemento sender serve al destinatario per capire da chi provenga il messaggio. Poiché mittente e destinatario si sono già accordati attraverso lo scambio dei messaggi CAP non è necessario che il sender esibisca nuovamente il certificato che attesta la sua identità, ma è sufficiente che fornisca il suo identificativo attraverso l'attributo obbligatorio **uniqueName.**

### *3.5 Elemento resource*

Nel caso la conversazione sia onResource o aboutResource questo elemento con attributo obbligatorio **uri** identifica dove si trova la risorsa a cui si riferisce l'espressione inviata.

### *3.6 Elemento content*

Questo elemento permette di specificare il formato dati del contenuto presente all'interno del body del messaggio SOAP. Questo elemento è opzionale poiché le parti si sono già accordate sul formato dati nel CAP.

# Appendice B: XML Schema

## 1 XML Schema del CAP

### proposal.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 targetNamespace="http://www.cs.unibo.it/CAP/1.0/proposal.xsd"
 xmlns="http://www.cs.unibo.it/CAP/1.0/proposal.xsd">

 <!--ELEMENTI-->

  <xs:element name="proposal" type="Tproposal"/>

  <xs:complexType name="Tproposal">
     <xs:sequence>
        <xs:element name="sender" type="TpartyInfo" maxOccurs="1"/>
        <xs:element name="request" type="Trequest" maxOccurs="1"/>
        <xs:element name="conversation" type="Tconversation"
         maxOccurs="unbounded"/>
        <xs:element name="security" type="Tsecurity" minOccurs="0"/>
     </xs:sequence>
     <xs:attribute  name="renegotiable" type="Treneg" use="required"/>
  </xs:complexType>

  <xs:complexType name="TpartyInfo">
     <xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="0"/>
        <xs:element name="certificate" type="Tcertificate"
         minOccurs="0"/>
     </xs:sequence>
     <xs:attribute name="uniqueName" type="xs:string" use="required"/>
  </xs:complexType>

  <xs:complexType name="Tcertificate">
     <xs:sequence>
        <xs:any namespace="##other" maxOccurs="1"
           processContents="lax"/>
     </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Trequest">
     <xs:sequence>
        <xs:element name="invited" type="Tinvited"/>
     </xs:sequence>
     <xs:attribute name="type" type="Ttype">
  </xs:complexType>
  <xs:complexType name="Tinvited">
     <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="party" type="TpartyInfo"/>
     </xs:sequence>
  </xs:complexType>
```

```
<xs:complexType name="Tconversation">
   <xs:sequence>
      <xs:choice>
         <xs:element name="onResource" type="TonRes"/>
         <xs:element name="isResource" type="TisRes"/>
         <xs:element name="aboutResource" type="TaboutRes"/>
      </xs:choice>
      <xs:element name="transport" type="Ttransport"/>
   </xs:sequence>
   <xs:attribute name="media" type="Tmedia" use="required"/>
   <xs:attribute name="id" type="xs:ID" use="required"/>
   <xs:attribute name="combineWith" type="xs:string"
    use="optional"/>
   <xs:attribute name="favorite" type="Tfavorite"/>
   <xs:attribute name="quality" type="Tquality"/>
</xs:complexType>

<xs:complexType name="TonRes">
   <xs:choice>
      <xs:element name="localLock" type="TlocalLock"/>
      <xs:element name="globalLock" type="TglobalLock"/>
      <xs:element name="noLock" type="TnoLock"/>
   </xs:choice>
   <xs:attribute name="uri" type="xs:anyURI" use="required"/>
</xs:complexType>

<xs:complexType name="TisRes">
   <xs:choice>
      <xs:element name="localLock" type="TlocalLock"/>
      <xs:element name="globalLock" type="TglobalLock"/>
      <xs:element name="noLock" type="TnoLock"/>
   </xs:choice>
</xs:complexType>

<xs:complexType name="TaboutRes">
   <xs:sequence>
      <xs:element name="noLock" type="TnoLock"/>
   </xs:sequence>
   <xs:attribute name="uri" type="xs:anyURI" use="required"/>
</xs:complexType>

<xs:complexType name="TlocalLock">
   <xs:choice>
      <xs:element name="multipleUtterance" type="TUtterance"/>
      <xs:element name="singleUtterance" type="TUtterance"/>
   </xs:choice>
   <xs:attribute name="grain" type="Tgrain" use="required"/>
   <xs:attribute name="token" type="Ttoken" use="optional"/>
</xs:complexType>

<xs:complexType name="TglobalLock">
   <xs:choice>
      <xs:element name="multipleUtterance" type="TUtterance"/>
       <xs:element name="singleUtterance" type="TUtterance"/>
    </xs:choice>
    <xs:attribute name="token" type="Ttoken" use="optional"/>
</xs:complexType>
```

```
  <xs:complexType name="TnoLock">
     <xs:sequence>
        <xs:element name="multipleUtterance" type="TUtterance"/>
     </xs:sequence>
  </xs:complexType>

  <xs:complexType name="TUtterance">
     <xs:attribute name="update" type="Tupdate" use="required"/>
     <xs:attribute name="visibility" type="Tvisibility"
      use="required"/>
  </xs:complexType>

  <xs:complexType name="Ttransport">
     <xs:sequence>
        <xs:element name="transportProtocol" type="TProtocol"
         maxOccurs="unbounded"/>
        <xs:element name="securityProtocol" type="TProtocol"
         minOccurs="0"/>
     </xs:sequence>
  </xs:complexType>

  <xs:complexType name="TProtocol" mixed="true">
     <xs:attribute name="version" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:complexType name="Tsecurity">
     <xs:sequence>
        <xs:any namespace="##other" minOccurs="0"
         processContents="lax"/>
        <xs:any namespace="##other" minOccurs="0"
         processContents="lax"/>
     </xs:sequence>
  </xs:complexType>


<!--ATTRIBUTI-->

  <xs:simpleType name="Treneg">
     <xs:restriction base="xs:boolean">
        <xs:pattern value="true|false"/>
     </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Ttype">
     <xs:restriction base="xs:string">
        <xs:enumeration value="createNew"/>
        <xs:enumeration value="participate"/>
     </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Tmedia">
     <xs:restriction base="xs:string">
        <xs:enumeration value="text/plain"/>
        <xs:enumeration value="text/html"/>
        <xs:enumeration value="text/xml"/>
        <xs:enumeration value="audio/mpeg"/>
        <xs:enumeration value="video/mpeg"/>
       <xs:enumeration value="video/quicktime"/>
        <xs:enumeration value="image/gif"/>
```

```
        <xs:enumeration value="image/jpeg"/>
        <xs:enumeration value="image/bmp"/>
        <xs:enumeration value="image/gif"/>
        <xs:enumeration value="application/octet-stream"/>
     </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Tfavorite">
     <xs:restriction base="xs:string">
         <xs:pattern value="true|false"/>
      </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Tquality">
     <xs:restriction base="xs:decimal">
        <xs:pattern value="1.00|0.80|0.50|0.30|0.00"/>
     </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Tgrain">
     <xs:restriction base="xs:string">
        <xs:enumeration value="character"/>
        <xs:enumeration value="word"/>
        <xs:enumeration value="line"/>
        <xs:enumeration value="paragraph"/>
        <xs:enumeration value="chapter"/>
        <xs:enumeration value="page"/>
      </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Ttoken">
     <xs:restriction base="xs:string">
        <xs:enumeration value="pool"/>
        <xs:enumeration value="client"/>
        <xs:enumeration value="server"/>
        <xs:enumeration value="user"/>
     </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Tupdate">
     <xs:restriction base="xs:string">
        <xs:enumeration value="pseudoPush"/>
        <xs:enumeration value="realPush"/>
        <xs:enumeration value="pull"/>
     </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Tvisibility">
     <xs:restriction base="xs:string">
        <xs:enumeration value="private"/>
        <xs:enumeration value="public"/>
     </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

## 2 XML Schema del CUML

### utterance.xsd

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 targetNamespace="http://www.cs.unibo.it/CUML/1.0/utterance.xsd"
 xmlns="http://www.cs.unibo.it/CUML/1.0/utterance.xsd">

<!--ELEMENTI-->

  <xs:element name="utterance" type="Tutterance"/>

  <xs:complexType name="Tutterance">
     <xs:sequence>
        <xs:element name="conversation" type="Tconversation"/>
        <xs:element name="sender" type="Tsender"/>
        <xs:element name="resource" type="Tresource"/>
        <xs:element name="content" type="Tcontent"/>
        <xs:element name="security" type="Tsecurity"/>
     </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Tconversation">
     <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>

  <xs:complexType name="Tsender">
     <xs:attribute name="uniqueName" type="xs:string" use="required"/>
  </xs:complexType>

  <xs:complexType name="Tresource">
     <xs:attribute name="uri" type="xs:anyURI" use="required"/>
  </xs:complexType>

  <xs:complexType name="Tcontent">
     <xs:attribute name="media" type="Tmedia" use="required"/>
  </xs:complexType>

  <xs:complexType name="Tsecurity">
     <xs:sequence>
        <xs:any namespace="##other" minOccurs="0"
         processContents="lax"/>
        <xs:any namespace="##other" minOccurs="0"
         processContents="lax"/>
     </xs:sequence>
  </xs:complexType>

<!--ATTRIBUTI-->

  <xs:simpleType name="Tmedia">
     <xs:restriction base="xs:string">
        <xs:enumeration value="text/plain"/>
        <xs:enumeration value="text/html"/>
```

```
        <xs:enumeration value="text/xml"/>
        <xs:enumeration value="audio/mpeg"/>
        <xs:enumeration value="video/mpeg"/>
        <xs:enumeration value="video/quicktime"/>
        <xs:enumeration value="image/gif"/>
        <xs:enumeration value="image/jpeg"/>
        <xs:enumeration value="image/bmp"/>
        <xs:enumeration value="image/gif"/>
        <xs:enumeration value="application/octet-stream"/>
      </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

# Appendice C: Esempi di firma e criptazione

Durante una sessione collaborativa, nel caso si voglia garantire confidenzialità, autenticazione e integrità dei documenti XML inviati è possibile utilizzare i meccanismi forniti dalle specifiche XML Digital Signature e XML Encryption descritte nel paragrafo 4.3.
E' possibile firmare e criptare sia i messaggi CAP che i messaggi CUML e in questo paragrafo, per mostrare un'esempio, verrà usato un messaggio CUML che sarà via via firmato, criptato e infine firmato e criptato.

## 1 Utterance firmata

Ecco la struttura di un messaggio CUML firmato:

```
<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
  http://schemas.xmlsoap.org/soap/envelope/" >

  <soap:header >
    <cuml:utterance id="utterance"
    xmlns:cuml="http://www.cs.unibo.it/CUML/1.0">

      <cuml:conversation id="..."/>
      <cuml:sender uniqueName="..."/>
      <cuml:resource uri="..."/>
      <cuml:content media="..."/>
      <cuml:Security
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
           <ds:SignedInfo>
              <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
              <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
               <ds:Reference URI="#utterance">
                   <ds:Transforms>
                       <ds:Transform
                 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                   </ds:Transforms>
                   <ds:DigestMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                      <ds:DigestValue>hONgCCP9OUOMk0p</ds:DigestValue>
              </ds:Reference>
```

```
            <ds:Reference URI="#myBody">
               <ds:Transforms>
                    <ds:Transform
               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
               </ds:Transforms>
               <ds:DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
               <ds:DigestValue>8Yqxg2jXETXZa2edDwph</ds:DigestValue>
            </ds:Reference>
         </ds:SignedInfo>
         <ds:SignatureValue>jUKqm16LAm..qHDrHnU...uOaEjEpfkhdsff0iNf
         </ds:SignatureValue>
         <ds:KeyInfo></ds:KeyInfo>
         </ds:Signature>
      </cuml:Security>
    </cuml:utterance>

  </soap:header>
  <soap:Body id="MyBody">
       <html>
           <head><title>esempio</title></head>
           <body><p>Hello world</p></body>
       </html>
  </soap:Body>
</soap:Envelope>
```

## 2 Utterance criptata

Ecco la struttura di un messaggio CUML criptato:

```
<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
 http://schemas.xmlsoap.org/soap/envelope/" >

 <soap:header >
   <cuml:utterance id="utterance"
    xmlns:cuml="http://www.cs.unibo.it/CUML/1.0">

         <cuml:conversation id="..."/>
         <cuml:sender uniqueName="..."/>
         <cuml:resource uri="..."/>
         <cuml:content media="..."/>
             <cuml:Security
               <enc:EncryptedKey
                xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
                 <enc:EncryptionMethod
              Algorithm=”http://www.w3.org/2001/04/xmlenc#rsa-1_5”/>
                 <KeyInfo
                   xmlns="http://www.w3.org/2000/09/xmldsig#"/>
                 </KeyInfo>
                 <CipherData
```

```
                    xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <CipherValue>bJd5Xk0g...fGkulBo23Unv</CipherValue
                  </CipherData>
                  <ReferenceList
                  xmlns="http://www.w3.org/2001/04/xmlenc#">
                     <DataReference URI="#enc"/>
                  </ReferenceList>
              </enc:EncryptedKey>
           </cuml:Security>
       </cuml:utterance >
   </soap:Header>
   <soap:Body id="body">
           <EncryptedData Id="enc"
             Type="http://www.w3.org/2001/04/xmlenc#Content"
             xmlns="http://www.w3.org/2001/04/xmlenc#">
               <EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
                 <CipherData>
                      <CipherValue>r+WpG9C1gYP..0+HVp5</CipherValue>
                 </CipherData>
           </EncryptedData>
   </soap:Body>
</soap:Envelope>
```

## 3 Utterance firmata e criptata

Ecco la struttura di un messaggio CUML firmato e criptato:

```
<?xml version="1.0" encoding="UTF-8"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
 http://schemas.xmlsoap.org/soap/envelope/" >

 <soap:header>
   <cuml:utterance id="utterance"
    xmlns:cuml="http://www.cs.unibo.it/CUML/1.0">
     <cuml:conversation id="..."/>
     <cap:sender uniqueName="..."/>
          <cuml:resource uri="..."/>
          <cuml:content media="..."/>
          <cuml:Security>
             <enc:EncryptedKey
               xmlns:enc="http://www.w3.org/2001/04/xmlenc#">
                <enc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
                   <KeyInfo
                    xmlns="http://www.w3.org/2000/09/xmldsig#">
                 </KeyInfo>
                 <CipherData
                   xmlns="http://www.w3.org/2001/04/xmlenc#">
                   <CipherValue>F3Hm0umLCx/8...Aa3m8b0p</CipherValue>
                 </CipherData>
```

```
                <ReferenceList
                  xmlns="http://www.w3.org/2001/04/xmlenc#">
                  <DataReference URI="#enc"/>
                </ReferenceList>
             </enc:EncryptedKey>
             <ds:Signature
               xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                   <ds:CanonicalizationMethod
               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                   <ds:SignatureMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                   <ds:Reference URI="#mybody">
                      <ds:Transforms>
                          <ds:Transform
               Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                          </ds:Transforms>
                          <ds:DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                               <ds:DigestValue>
                                  kNY6X7LnRTwxXXBzSw07tcA0KSU=
                               </ds:DigestValue>
                   </ds:Reference>
                 </ds:SignedInfo>
                 <ds:SignatureValue>
                    9j3fHRnClcE...MzP06ge...j2YPD8O4TfayGvuYglLfwuhr
                 </ds:SignatureValue>
                 <ds:KeyInfo></ds:KeyInfo>
             </ds:Signature>
       </cuml:Security>
     </cuml:utterance>
  </soap:Header>
  <soap:Body id="Mybody>
          <EncryptedData Id="enc"
             Type="http://www.w3.org/2001/04/xmlenc#Content"
             xmlns="http://www.w3.org/2001/04/xmlenc#">
                <EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
                <CipherData>
                    <CipherValue>
                       tjOgUPMmQwd6hXiH...oxs1/MWb0o...G+kTvNrtgjPje
                    </CipherValue>
                </CipherData>
          </EncryptedData>
  </soap:Body>
</soap:Envelope>
```