

**Il Progetto SAPI**  
**(Sistema Automatico Per Ipovedenti)**

# Il Progetto SAPI (Sistema Automatico Per Ipovedenti)

Pina Russo [russog1@posteitaliane.it](mailto:russog1@posteitaliane.it) (1), Eugenio Manfredonia [manfredoniae@posteitaliane.it](mailto:manfredoniae@posteitaliane.it) (1), Alfredo Troiano [alfredo.troiano@its.na.it](mailto:alfredo.troiano@its.na.it) (2), Francesco Gragnani [f.gragnani@criai.it](mailto:f.gragnani@criai.it) (3)  
(1) Poste italiane, (2) ITS spa, (3) CRIAI scarl

## Abstract

Obiettivo del progetto SAPI è la realizzazione di una piattaforma software per l'erogazione di contenuti e servizi, con particolari ed innovative caratteristiche di adattività, accessibilità, flessibilità, multimodalità ed indipendenza dal contesto di riferimento. Il progetto SAPI rappresenta la prima esperienza "fully" open source di Poste italiane.

## Keywords

Adattamento, Contenuti Intelligenti, Contesto, Ipovedenti, Ontologie, Servizi Intelligenti.

## 1. Scopo del progetto

Il progetto si rivolge a tutti gli utenti fruitori di servizi e contenuti, ma, in maniera particolare ad un'utenza diversamente abile e principalmente a quella ipovedente e non vedente.

La realizzazione della piattaforma SAPI giunge al termine di una approfondita attività di ricerca e di definizione di modelli, metodologie e tecniche basate sulla conoscenza, finalizzate alla personalizzazione dell'interazione uomo-macchina in contesti di e-business con particolare (ma non esclusivo) riferimento all'erogazione di contenuti e servizi on-line ed al supporto all'accessibilità nelle interazioni con soggetti ipovedenti.

Il progetto si pone l'obiettivo di adattare l'erogazione di contenuti e servizi sulla base di differenti parametri tra i quali: il profilo dell'utente e le sue abilità, il dispositivo di accesso, il canale trasmissivo e le caratteristiche del luogo fisico nel momento in cui viene istanziata la comunicazione.

Oltre alla realizzazione della piattaforma di erogazione, il progetto vuole anche approfondire la ricerca sui *contenuti intelligenti* [1], che postula la realizzazione di contenuti in grado di auto-adattarsi rispetto al profilo dell'utente ed al contesto di fruizione, estendendo tale concetto anche ai servizi definendo *servizi intelligenti ed adattivi* quelli in grado di adattare la propria interazione con l'utente finale sulla base di modelli di interazione formalmente definiti. Servizi e contenuti intelligenti saranno in grado di reagire in maniera differente a seconda dell'utente che interagisce (considerandone sia il profilo che le abilità) e del contesto di interazione (ivi includendo il dispositivo di accesso, il canale trasmissivo e le caratteristiche ambientali).

## 2. Stato di avanzamento

Il progetto ha visto una prima fase di analisi dello stato dell'arte, dal punto di vista delle norme di riferimento, da quello degli ausili hardware/software per utenza ipovedente e non vedente, e da quello, più specificatamente infrastrutturale, di progetti ed esperienze nazionali ed internazionali di piattaforme adattive e di servizi e contenuti intelligenti. Successivamente si è passati alla definizione dell'architettura funzionale ed allo studio dei modelli per l'utente, il contesto e dominio dell'e-business.

Definita l'architettura funzionale della piattaforma è stato effettuata una mappatura tra i blocchi funzionali ed i possibili progetti Open Source che potessero essere utilizzati a copertura dei blocchi stessi.

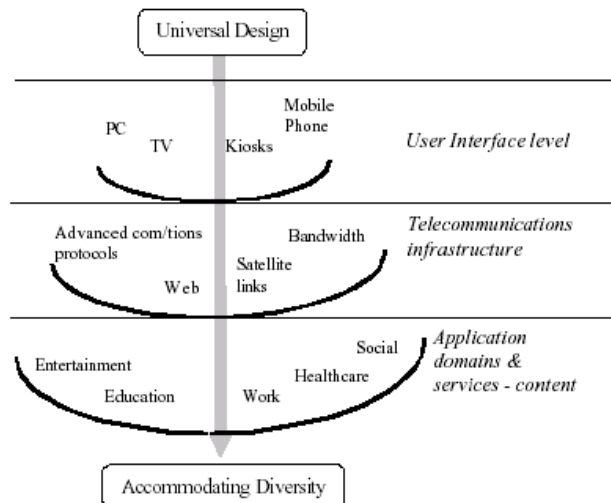
Si è così giunti ad una lista di progetti la cui integrazione costituirà il core della piattaforma SAPI.

Attualmente, gran parte delle scelte effettuate ha carattere definitivo, ed il progetto è nella fase di risoluzione delle difficoltà nella comunicazione ed interoperabilità tra i differenti moduli componenti la piattaforma.

### 3. Architettura di riferimento

#### 3.1. Approccio architetturale

L'approccio adottato in SAPI ha seguito i principi dello "Universal Design" che sostituisce al concetto di utente medio la considerazione delle diverse abilità, le richieste e le preferenze dell'utente, fin dalle prime fasi della progettazione e dello sviluppo secondo una realizzazione proattiva.



**Modello di Universal Design**

Nel definire il modello architetturale di SAPI è stata data primaria importanza ai seguenti drivers architetturali:

- **User Context awareness:** SAPI è una piattaforma in grado di "adattarsi", di modificare il proprio comportamento e le informazioni fornite, in base al *contesto* nel quale si trova l'utente, al fine di soddisfare al meglio le sue necessità e le sue esigenze;
- **Decentralizzazione dell'intelligenza nelle entità di SAPI auto-personalizzabili:** rendere il più possibile autonomi i Servizi intelligenti e i Contenuti intelligenti per quanto concerne l'adattività al contesto dell'utente;
- **Adattamento centralizzato** delle entità di SAPI non autonomamente personalizzabili (approccio tradizionale);
- **Fruizione "ubiqua" di servizi e contenuti** che presentino caratteristiche di "sensibilità al contesto d'utente".

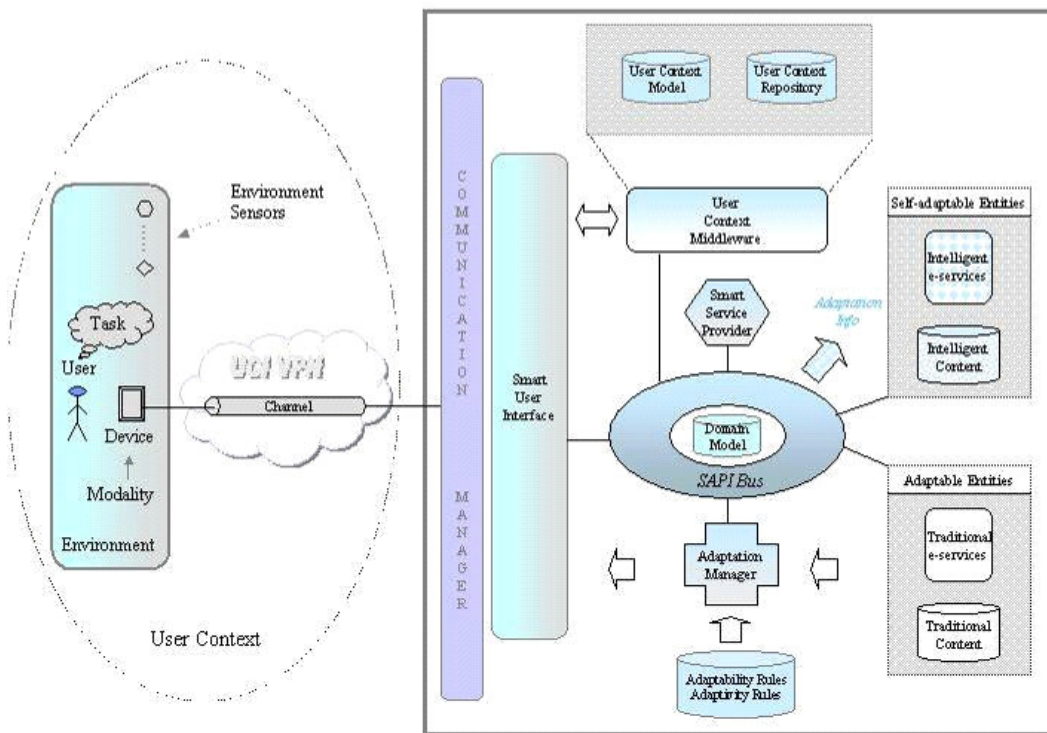
#### 3.2. Architettura funzionale

L'architettura nasce con una struttura che prevede blocchi funzionali di base (building blocks). Tali blocchi sono interconnessi così che, attraverso la loro cooperazione ed il conseguente scambio di flussi informativi, si possano realizzare tutte le funzionalità previste.

Attraverso la decomposizione dei blocchi in entità funzionali, risulta più agevole fornire una vista di insieme dell'architettura funzionale di SAPI che riporti da un lato la scomposizione funzionale della piattaforma, dall'altro i legami funzionali esistenti tra le varie componenti.

Una delle caratteristiche peculiari di un sistema ubiquo e pervasivo è l'estrema eterogeneità e complessità del contesto d'uso in cui si trova l'utente, costituito dall'ambiente, dalle varie entità presenti nell'ambiente ed in particolare dai dispositivi e dalle connessioni (canale di comunicazione) che gli utenti possono usare per accedere ai servizi disponibili, in un dato dominio applicativo, nella modalità a loro più congeniale. La piattaforma SAPI si propone di modellare, seguendo un approccio ontologico, lo User Context (integrazione dello User Model e del Context Model) e il Dominio Applicativo (Domain Model) allocando le proprietà

relative ai singoli concetti dello User Context in uno spazio multidimensionale (User Context Space).



### Principali componenti funzionali del Framework SAPI

La definizione e la rappresentazione ontologica dello User Context consentirà di avere un'interpretazione sintattica e semantica condivisa da chi acquisisce e gestisce l'informazione contestuale da un lato (User Context Middleware) e da chi la utilizza (Contenuti e Servizi Intelligenti, motore di adattamento) dall'altro.

Analizziamo i blocchi dell'architettura un po' più in dettaglio, con riferimento ai successivi paragrafi nei quali saranno evidenziate le soluzioni Open Source selezionate per l'implementazione.

#### 3.2.1. User Context Middleware

Prevede di gestire, attraverso una opportuna logica, basi di conoscenza contenenti informazioni su contesto, utente, dominio (User, Context e Domain Model) e di renderle disponibili alle altre entità SAPI. Vista la forte dinamicità e rapidità di evoluzione, la frequenza di aggiornamento dello User Model sarà molto maggiore rispetto a quella degli altri due.

Con riferimento alla individuazione delle soluzioni, occorre focalizzare l'attenzione su:

- logica di gestione di basi di conoscenza e di regole di evoluzione dei modelli
- logica di gestione del contesto

#### 3.2.2 Adaptation Manager, Entità autopersonalizzabili

Entrambi i macro-blocchi devono poter eseguire logiche di adattamento sia dal punto di vista dei servizi sia dal punto di vista dei contenuti, in base allo user-context associato alla richiesta dell'utente. Ciò introduce la necessità di prevedere una gestione dell'adattamento.

### 3.2.3 Communication Manager

E' stato previsto per garantire la sicurezza dell'accesso alla piattaforma, l'autenticazione dell'utente e la privacy. Allo scopo deve essere dedicato un blocco di gestione dell'accesso alla piattaforma.

### 3.2.4. Smart Service Provider, SAPI Bus

Lo Smart Service Provider deve soddisfare le richieste dell'utente orchestrando opportunamente i servizi disponibili. Il SAPI bus deve garantire lo scambio di informazioni tra tutte le entità SAPI. Tali requisiti si sposano con quanto previsto dall'approccio SOA (Service Oriented Architecture) che, pertanto, sarà adottato.

## 4. Metodologie realizzative dei macro-blocchi funzionali

Per la realizzazione dei macro-blocchi funzionali presenti nell'architettura di riferimento, di seguito vengono descritte le soluzioni da adottare ai fini della loro successiva realizzazione. Tali soluzioni coprono la quasi totalità dei macro-blocchi funzionali, lasciando alla progettazione e allo sviluppo di codice sorgente appositamente scritto in Java il compito di realizzare eventuali integrazioni, comunicazioni e orchestrazioni dei framework esistenti e utilizzati come componenti logici.

A tale proposito, la scelta del linguaggio Java per lo sviluppo di codice sorgente, oltre a garantire la portabilità sui sistemi operativi, facilita l'integrazione di codice customizzato con l'ampio numero di strumenti open source disponibili. Il linguaggio Java mette a disposizione, inoltre, il meccanismo Java Native Interface (JNI), per richiamare codice C all'interno di oggetti Java. Ciò permette di sviluppare, all'occorrenza, codice di più basso livello quando si devono interfacciare driver non supportati nativamente da Java.

### 4.1 Gestione della base di conoscenza e delle regole di evoluzione dei modelli

Nell'ambito del progetto SAPI viene considerato il formalismo delle ontologie [2] al fine di ottenere la più ampia compatibilità con le iniziative legate al Semantic Web [3]. Il linguaggio di riferimento è, pertanto, OWL [4] (con le iniziative correlate: XML per la serializzazione, OWL-QL per l'interrogazione, ecc.). Peraltro, la fase di modellazione prevede, come apporto originale del progetto, l'inserimento nelle ontologie di relazioni sia **esplicite** (definite dall'esperto del dominio) che **implicite** (ricavate dal sistema). Al fine di consentire questa estensione verranno considerati sia approcci probabilistici (e.g. reti Bayesiane [5] che possibilistici (e.g. logiche fuzzy [6])

Oltre al formalismo delle ontologie, il progetto SAPI non trascurava l'approfondimento di altre tecniche di modellazione Knowledge-based (stereotipi, profili rule-based, etc.). A tale scopo è stato necessario adottare strumenti software sufficientemente flessibili e, nel contempo, facilmente integrabili con altro codice sorgente progettato e sviluppato all'occorrenza.

A supportare la diffusione e l'utilizzo delle ontologie presso tutti i possibili utenti è da evidenziare il contributo apportato dal mondo dell'open source nel mettere a punto soluzioni sempre più complete, che si adeguino alle raccomandazioni W3C, che consentano l'utilizzo completo dei linguaggi ontologici (ad esempio OWL) e che introducano interfacce grafiche user-friendly per evitare di vincolare la realizzazione delle ontologie ad una nicchia prescelta di progettisti

Per la definizione delle metodologie di rappresentazione delle regole di evoluzione dei modelli (reti bayesiane, automi a stati finiti, ecc.) basate anche sulla storia delle interazioni degli utenti con la piattaforma occorrerà una ulteriore fase di studio. Le scelte eseguite a valle di tale studio serviranno ai progettisti di SAPI per integrare gli strumenti per la gestione della base di conoscenza con *reasoner* esistenti e/o opportunamente customizzati al fine di gestire l'evoluzione dei modelli.

### 4.2 Gestione dell'adattamento

Nel corso del progetto si intende sperimentare la personalizzazione, in riferimento a quanto proposto in [7], [8], al fine di assicurare un alto livello di qualità dell'interazione con il sistema a tutti i potenziali utenti (abili e

diversamente abili).

La piattaforma SAPI deve adattare il suo comportamento alle caratteristiche dell'utente oggettive (abilità, bisogni, interessi, preferenze) e soggettive (comportamento), al contesto d'uso (ambiente, terminale, rete, periferiche speciali), nonché allo stato corrente dell'interazione e alla storia precedente.

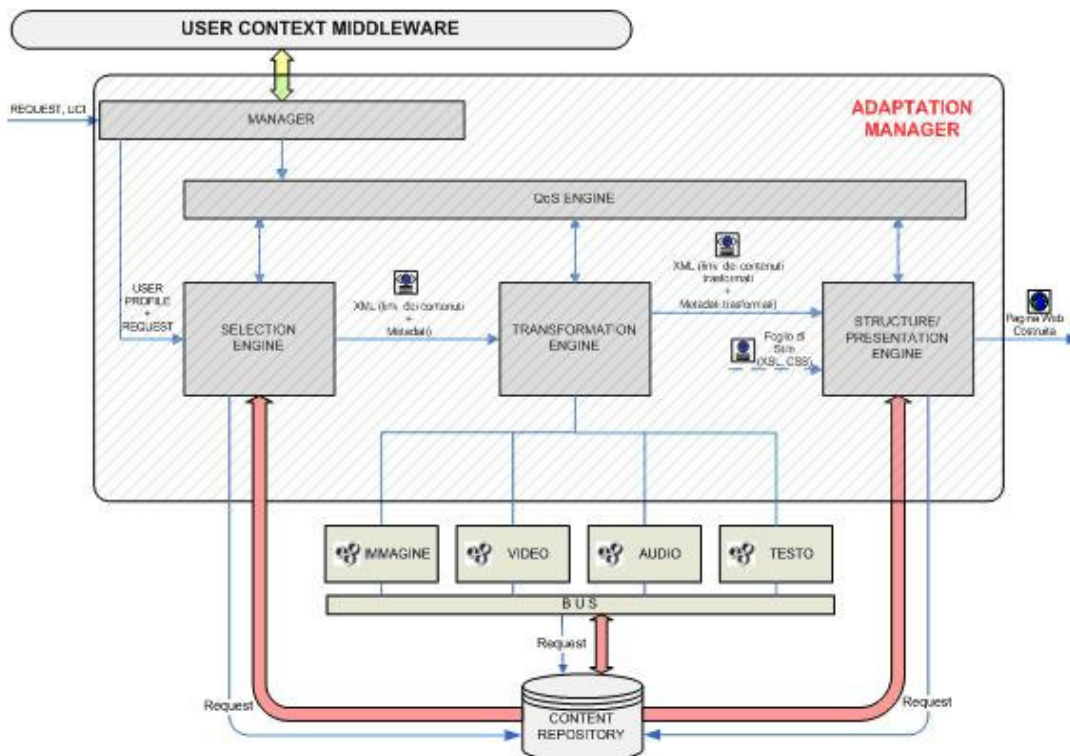
Il concetto di adattamento (adaptation) può essere scomposto nei concetti di adattabilità (adaptability) e adattività (adaptivity). L'adattabilità si riferisce al processo di selezionare/modificare l'espressione della user interface durante la fase iniziale di ciascuna sessione di interazione, in accordo alle caratteristiche dell'utente e del contesto d'uso che sono note prima dell'interazione (ad es. user abilities) e sono assunte rimanere invariate durante la singola sessione (ad. es. particolare user expertise).

L'adattività si riferisce al processo di selezionare/modificare l'espressione della user interface dinamicamente, in accordo alle caratteristiche dinamiche dell'utente, del relativo contesto d'uso, e di situazioni rilevabili a run-time (ad es. alto tasso di errore, incapacità a completare un task, ecc.).

A partire dai concetti di adattabilità e adattività si può definire il concetto di personalizzazione analizzato in SAPI: "Un Sistema Informativo Personalizzabile è un Sistema Informativo (in quanto fornisce contenuti informativi e servizi) adattabile sulla base delle caratteristiche peculiari degli utenti, quali interessi, preferenze, comportamenti e delle situazioni (contesti) d'utente, e che tende a massimizzare la user experience degli utenti finali durante lo svolgimento dei loro task".

In SAPI l'adattamento [9] sarà sperimentato sia in modalità distribuita (arricchendo di apposita logica di gestione dell'adattamento i servizi e i contenuti) sia in modalità centralizzata (prevedendo una logica di gestione dell'adattamento applicabile a servizi e contenuti personalizzabili, ma già esistenti).

Nella figura sottostante viene dettagliata la realizzazione della gestione dell'adattamento in SAPI relativamente a QoS, contenuti e presentazione.



**La Gestione dell'Adattamento in SAPI**

Relativamente all'adattamento del contenuto, si svilupperà un adattamento di tipo ibrido, cioè basato sia sulla transcodifica (in tempo reale) delle proprietà del contenuto sia sulla selezione del contenuto più appropriato (tra le varie versioni disponibili) per soddisfare la richiesta dell'utente, vedi [10].

Relativamente all'adattamento della funzionalità, è in fase di studio l'adozione di un modello che faccia riferimento alle Entità Astratte di Interazione (Abstract Interaction Units) già proposte in [11] per la descrizione dei servizi previsti. Tale studio consentirà di sviluppare l'appropriata logica di gestione di adattamento rispetto alle funzionalità esportate dai servizi.

### 4.3 Gestione del contesto

I sistemi mobili operano in un contesto molto variabile, la cui causa è da attribuire principalmente a due fattori: la rete e i dispositivi.

Questa intrinseca "instabilità" del contesto, giustifica la necessità di realizzare servizi adattativi e il paradigma degli agenti mobili rappresenta una naturale soluzione alla realizzazione di tali servizi [12].

Peraltro, in presenza di reti fortemente dinamiche con una molteplicità di dispositivi che si agganciano e sganciano dalla rete o migrano continuamente (PDA, smartPhone, cellulari 3G, ecc.), è preferibile che gli stessi elementi di controllo siano in grado di migrare al fine di rimodulare le aree di propria competenza. Risultano infatti insufficienti gli approcci tradizionali di monitoraggio del contesto.

Da tali considerazioni risulta opportuno che i componenti coinvolti nel controllo godano anche di una certa autonomia e capacità propria.

Avvalora ulteriormente tale scelta la necessità di studiare il comportamento dell'utente durante l'interazione (usando una particolare tipologia di sensori tuttora in fase di progettazione, detti "virtuali").

La risposta a queste esigenze è rappresentata appunto dal paradigma degli agenti mobili, i quali presentano le caratteristiche di dinamicità, autonomia e reattività richieste.

La gestione del contesto passa pertanto attraverso la realizzazione di un "Multi Agent System" (MAS).

### 4.4 Gestione della comunicazione tra entità SAPI

Si prevede lo studio e la sperimentazione di un'architettura tecnologica ibrida "popolata" da Agenti e Servizi tra loro cooperanti nella realizzazione delle funzionalità. A seconda della funzionalità richiesta si possono utilizzare sia Agenti che Servizi al fine di soddisfare l'obiettivo. Ad esempio, potrebbe essere un Agente l'entità funzionale che sulla base dei valori acquisiti dai sensori luminosità e rumore ambientale regola la luminosità dello schermo e il volume dell' audio di un chiosco multimediale. Da queste considerazioni si evince che in SAPI dovranno coesistere e comunicare tra loro entità eterogenee che, parlando linguaggi differenti (ACL, WSDL,...), avranno bisogno di un interprete (gateway). Tale gateway è stato individuato nel SAPI Bus, che ha il compito di virtualizzare i vari linguaggi fornendo un linguaggio comune (SAPI Bus API) di comunicazione alle varie entità di SAPI. Verso il mondo esterno al dominio SAPI le Entità SAPI comunicheranno attraverso meccanismi di comunicazione firewall friendly standard (ad es. WSDL/XML/SOAP su HTTP) per garantire interoperabilità.

Nella piattaforma SAPI è previsto il concetto di adattamento al dominio dell'utente e al contesto di interazione. Essa offre una serie di servizi che possono essere sia autoadattabili (servizi intelligenti) che soggetti a un workflow per l'adattamento (servizi non intelligenti, ma personalizzabili).

In particolare, l'entità che esegue l'adattamento richiede attraverso il SAPI Bus allo User Context Middleware (UCM) l'istanza di User Context e la User Context History rispetto alle quali dovrà eseguire tecniche di adattamento. Inoltre, richiederà al Domain Model Manager (DMM) le informazioni di dominio pertinenti. Dal punto di vista della loro definizione e gestione i servizi intelligenti di SAPI saranno conformi al paradigma architetturale SOA [14].

Quindi, l'idea è quella di realizzare una architettura orientata ai servizi (SOA) in cui coesistano e cooperino sia servizi realizzati da agenti che da Web Service. Infine il SAPI BUS avrà una stretta interazione con la Smart Service Provider (SSP) che consentirà l'orchestrazione dei vari servizi.

## 5. I componenti Open Source selezionati

Anche nel progetto SAPI l'adozione del paradigma Open Source comporta il frequente ricorso a tecnologie (DBMS, Sistemi Operativi, Web Server, Application Container, ecc.) molto note sia tra i progettisti e gli sviluppatori sia nel mondo accademico e della ricerca. Nel corso del presente paragrafo non ci si soffermerà in dettaglio su tali tecnologie rimandando all'abbondante letteratura sull'argomento. Nel seguito verranno viceversa descritti per esteso i componenti individuati e ritenuti indispensabili alla realizzazione della piattaforma.

## 5.1 Gestione della base di conoscenza e delle regole di evoluzione dei modelli

Dal punto di vista dello strumento di gestione della base di conoscenza la scelta è caduta su jena. I principali altri strumenti per la gestione di ontologie, infatti, o sono basati su tecnologia proprietaria (Cerebra, Autonomy) o si sono evoluti verso di essa (Kaon, Sesame). La versione open source di questi ultimi, peraltro, appare non più mantenuta.

Il framework Jena consente la generazione di modelli di dati in memoria persistente lasciando ampia possibilità di scelta tra numerosi database (MySQL, HSQLDB, Apache Derby, PostgreSQL, nonché altre soluzioni proprietarie).

Jena, oltre ad aver introdotto, a partire dalla release 2 il supporto per OWL-DL, fa capo ai laboratori di ricerca HP dove la folta comunità di sviluppatori fornisce il proprio supporto mediante apposita mailing-list. Jena implementa il motore inferenziale con un reasoner offrendone diverse tipologie: il GenericRuleReasoner, il reasoner RDF e il reasoner OWL. Il motore inferenziale di Jena è applicabile sia ad ontologie descritte in RDF che ad ontologie descritte in OWL.

Il motore inferenziale di Jena offre anche la possibilità di compiere l'operazione di validazione dei dati, ovvero un controllo di consistenza tra i valori delle istanze e quanto per essi specificato negli schema.

Jena risulta facilmente integrabile anche con altri software ragionatori messi a disposizione dalla comunità open source (ad esempio, Pellet, FaCT++). La possibilità di dedurre informazione implicita dai dati ha fortemente stimolato l'interesse verso l'utilizzo del Framework Jena nell'ambito della sperimentazione e dell'applicazione sviluppata.

L'uso di ontologie per la rappresentazione di un dominio di conoscenza si prefigge l'obiettivo di esprimere il significato inteso dei termini di un vocabolario condiviso, pertanto, la possibilità di utilizzare un motore inferenziale su una base di conoscenza permette di sfruttare la semantica dei dati introdotta dalle ontologie per dedurre nuove informazioni.

L'ontology design è realizzato, invece, con il tool Protégé [15]: un tool open source sviluppato presso l'Università di Stanford, che fornisce un editor grafico e interattivo, che consente di modificare e accedere velocemente alle ontologie, nonché di interagire direttamente con l'ontologia navigando l'albero che la rappresenta.

## 5.2 Gestione dell'adattamento

Rispetto alla gestione dell'adattamento prevista nel precedente paragrafo, i riferimenti open source disponibili in letteratura non consentono di coprirne tutta la logica. Pertanto, gran parte deve essere progettata e realizzata lavorando direttamente sul codice sorgente.

Tra i componenti utilizzabili a tale scopo saranno usati:

Gaia Image Transcoder [16]: libreria open source che opera la transcodifica di immagini dedicandosi particolarmente a dispositivi mobili utilizzando informazioni reperite dalla descrizioni dei dispositivi WURFL [17].

Image Magick [18]: una raccolta di programmi che consentono di identificare, modificare, convertire e visualizzare immagini.

## 5.3 Gestione del contesto

La gestione del contesto si baserà su di un sistema MAS specializzato nella rilevazione del contesto. Si dovrà creare una società di agenti e concretizzare i loro comportamenti perché siano capaci di interagire con i componenti di rete e con il dispositivo usato dall'utente.

I sistemi multi-agente sono software molto complessi, quindi la loro costruzione necessita di un approccio metodico. Per questo motivo si è deciso di scegliere un framework che aderisse ad uno standard e adottasse una ben precisa metodologia.

Gli standard di riferimento sono le specifiche FIPA. La scelta è ricaduta sull'utilizzo di JADE, che presenta anche implementazioni specifiche all'utilizzo da dispositivi mobili.

JADE [19] è un ambiente di sviluppo Java per agenti. Esso è un middleware per sistemi multi-agente, fornisce un'infrastruttura software e lo sviluppatore deve implementare solo alcuni aspetti dell'applicazione.

In tale sistema andranno opportunamente integrati software per l'acquisizione di dati grezzi e per fornire dati strutturati alle applicazioni che ne fanno uso.

A tale scopo si farà riferimento ai seguenti componenti Open Source:

Context-Toolkit (sviluppato al Georgia Tech's GVU) è un insieme di librerie scritte in Java che consentono di realizzare un'architettura per il processo di acquisizione del contesto dall'uso e delivery del contesto.

CMU Sphinx (progetto della Carnegie Mellon University in collaborazione con SUN) è un sistema di speech recognition interamente scritto in Java

DELI (DELivery context LIbrary): libreria open source sviluppata dagli HP Labs che permette a servlet Java di risolvere richieste http contenenti descrizioni di device in formato CC/PP (Composite Capabilities / Preferences Profile) o UAProf (User Agent Profile) e di interrogare i profili ottenuti

## 5.4 Gestione della comunicazione tra entità SAPI

Il compito del SAPI Bus è quello di virtualizzare i vari linguaggi fornendo un linguaggio comune di comunicazione alle varie entità che compongono il sistema. Proprio in questo contesto dove è richiesta l'integrazione di entità software diverse, l'adozione dell'Open Source trova il maggior riscontro.

In SAPI si prevede l'utilizzo combinato di Webservice e di Agenti, implementando una struttura complessa di WSIGS (Web Service Integration Gateway Service) per la realizzazione del SAPI Bus. In particolare in SAPI si sta valutando la soluzione offerta da IONA (CELTIX), una versione Open Source dell'Enterprise Service Bus commerciale ARTIX e quella proposta da LogicBlaze (ServerMix), un prodotto che è il primo ESB JBI-Compliant.

Dal punto di vista della loro definizione e gestione i servizi intelligenti e non intelligenti ma adattabili di SAPI saranno conformi al paradigma architetturale SOA. Ulteriore studio è previsto prima di passare alla loro progettazione e realizzazione.

Analogo sforzo è richiesto nell'ottica della realizzazione dello Smart Service Provider che è il modulo che "orchestra" l'erogazione dei servizi, ovvero, si occupa di legare in maniera intelligente più Servizi elementari, eventualmente arricchiti con descrizioni ontologiche, definendo dei workflow su di essi. Tale modulo potrebbe essere realizzato sia come agente software sia come servizio.

## 5.5 Gestione dell'accesso alla piattaforma

La piattaforma SAPI si vuole proporre come progetto pilota per sperimentare la possibilità di far accedere gli utenti seguendo le indicazioni previste dallo standard ETSI UCI (Universal Communications Identifier) [13].

Il concetto di VPN UCI è la soluzione che SAPI propone per:

- identificare e indirizzare univocamente l'utente in modo trusted durante le interazioni con il sistema informativo;
- codificare e gestire le caratteristiche oggettive del profilo di utente (esplicito) relative alle sue necessità (dovute ad un'eventuale disabilità visiva) e preferenze (tipo di device utilizzato, lingua, etc.) di comunicazione;
- trasmettere nell'infrastruttura di comunicazione in modo sicuro e privato le informazioni di utente.

Per uniformarsi al modello proposto da ETSI, la gestione dell'accesso alla piattaforma SAPI deve prevedere l'utilizzo di un sistema ad agenti per istanziare le entità previste dallo standard (Personal User Agent, che tiene conto del profilo utente, e Service Agent, che accede alla descrizione del servizio). Inoltre, risulta necessario prevedere un sistema SOA compliant per rendere disponibili i servizi all'utente.

Tutte le tecnologie necessarie a garantire la sicurezza nell'accesso sono ampiamente garantite da strumenti di larga diffusione nel mondo Open Source, quali Sistemi Operativi (Linux, nelle varie distribuzioni, FreeBSD, Slackware, ecc.), application container (Tomcat, Jboss, ecc.), web server (Apache, Roxen, ecc.).

In ogni caso la parte di connessione tra l'utente e il server di autenticazione (AS) deve essere protetta in

quanto i dati viaggiano sulla internet pubblica. Pertanto la VPN non dovrà interessare esclusivamente l'ambiente UCI e l'insieme delle sue entità, ma dovrà essere opportunamente estesa per inglobare l'utente stesso.

## 6. Conclusioni

Il progetto di ricerca SAPI, rivolto allo studio e alla sperimentazione di Sistemi Informativi Innovativi e Personalizzabili, vede nell'impiego di software Open Source una scelta necessaria ed imprescindibile.

L'approccio con spirito innovativo ed originale alla realizzazione di un framework software complesso, ha come prerequisito la possibilità di avere sotto controllo gli strumenti utilizzati. La possibilità di accedere al codice sorgente consente di studiarne il funzionamento e di apportarvi le modifiche necessarie.

Le tecnologie open source individuate hanno offerto software sul quale progettare gran parte delle soluzioni adottate per realizzare il prototipo previsto dal progetto SAPI.

In questo scenario l'esperienza fatta con il progetto SAPI rappresenta un punto di partenza ed un riferimento sia per i successivi progetti di ricerca, che per l'adozione di piattaforme per lo sviluppo di prodotti e servizi implementati direttamente dal gruppo di lavoro di Poste Italiane.

## 7. Riferimenti bibliografici

- [1] <http://www.acemedia.org/aceMedia/project/index.html>
- [2] Fensel D. *Ontologies: A Silver bullet for Knowledge Management and Electronic Commerci*. Springer-Verlag, 2000.
- [3] Michael C. Daconta, et al. *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. John Wiley & Sons; (2003).
- [4] OWL Web Ontology Language Overview. W3C Proposed Recommendation 15 Dec 2003. Deborah L. McGuinness and Frank van Harmelen eds.
- [5] D. Heckerman, D. Geiger, D. Chickering. *Learning Bayesian networks: The Combination of Knowledge and Statistical Data*. Machine Learning, 20:197-243, 1995.
- [6] LA Zadeh. *Fuzzy Logic*. Computer, 1988.
- [7] Van Setten, M. (2001). *Personalised Information Systems: Analysis of existing personalized information systems*. Giga)E/D2.3 and Duine/D1.1, Ensc(ede: Telematica Instituut (TI/RS/2001/036).
- [8] Adomavicius G. and A. Tuzhilin. 2001. *Using Data Mining Methods to Build Customer Profiles*. IEEE Computer, 34 (2), 74-82.
- [9] Biemans, M. C. M., van Setten, M., van Vliet, H., Alberink, M., Eertink, H., de Heer, J., van Kranenburg, H., Kruse, H., de Poot, H., Reitsma, J., Slagter, R., Teeuw, W., & Veenstra, M. (2002). *Adaptation. An integrated view on adaptation in telematics*, Telematica Instituut.
- [10] Chung-Sheng Li, Rakesh Mohan and John R. Smith, "Multimedia Content Description in the InfoPyramid," IEEE Proc. Int. Conf. Acoust., Speech, Signal Processing (ICASSP), Seattle, WA, Special session on Signal Processing in Modern Multimedia Standards, May, 1998
- [11] G. Ausiello, E. Bertini, A. Cali, T. Catarci, S. Kimani, G. Cantucci. *Designing Adaptable Multi-device Applications*. Progetto MAIS (Multi-channel Adaptive Information Systems), Fondo FIRB, Programma Strategico Tecnologie abilitanti per la Società della Conoscenza-ICT. [http://black.elet.polimi.it/mais/documenti\\_pubblico/IIsemestre/r7.3.2.pdf](http://black.elet.polimi.it/mais/documenti_pubblico/IIsemestre/r7.3.2.pdf).
- [12] G.Ventre et alii (CRIAL), *DICAMS: Un Middleware MultiAgente per Applicazioni Context-Aware*, 2006
- [13] European Telecommunications Standards Institute (ETSI), *Universal Communications Identifier (UCI); System framework*, [portal.etsi.org/docbox/EC\\_Files/EC\\_Files/eg\\_202067v010101p.pdf](portal.etsi.org/docbox/EC_Files/EC_Files/eg_202067v010101p.pdf)
- [14] Eric Newcomer, Greg Lomow, *Understanding SOA with Web Services*. Addison Wesley (2005).
- [15] <http://protege.stanford.edu/>
- [16] <http://gaia-git.sourceforge.net>
- [17] [wurfl.sourceforge.net](http://wurfl.sourceforge.net)
- [18] [www.imagemagick.org](http://www.imagemagick.org)
- [19] <http://jade.tilab.com/>