

## Metodologia di generazione dell'interfaccia utente Survey

Nel presente documento si riassumono i contenuti del deliverable "**D3.4\_1 Metodologia di generazione dell'interfaccia utente**" relativo all'attività *RI 3.4 Metodologie basate sulla conoscenza per la personalizzazione dell'interazione*, nell'ambito del terzo Obiettivo Realizzativo (OR 3) "Studio di modelli di descrizione formale dell'utente e del contesto e regole di evoluzione".

La *Human Computer Interaction* (HCI), nata da poco più di dieci anni, si occupa dei problemi connessi alla progettazione di interfacce uomo-macchina, cercando di offrire utili strategie e suggerimenti nel tentativo di rendere possibile un'efficace interazione fra l'utente ed il computer. Più che una vera e propria disciplina, costituisce un ambito interdisciplinare di ricerca. I fattori che si considerano sono la usabilità, l'accessibilità e l'ergonomia, che determinano le costrizioni del design dei sistemi e suggeriscono specifiche linee guida e standard da osservare in fase di progettazione. In tale processo di progettazione sono coinvolti principalmente tre "elementi": l'uomo, il computer e l'attività da svolgere. Ma l'evoluzione che si vuole raggiungere non riguarda esclusivamente la capacità di un computer di svolgere determinate operazioni, piuttosto il tipo di interazione che coinvolge l'utente e il sistema. A tale scopo è stato introdotto il concetto di *multimodalità* (quando un qualsiasi tipo di interazione coinvolge più di un canale percettivo o input di comunicazione) e pertanto si parlerà di *interfacce multimodali*. Ma ad ottenere vantaggi da un miglioramento della qualità dell'interazione fra uomo e macchina, non sarebbero soltanto le persone poco esperte che riuscirebbero ad utilizzare il computer con più facilità: si pensi piuttosto alle persone disabili, che pur conoscendo molto bene gli standard attuali, hanno problemi di tipo fisico che impediscono loro di interagire correttamente con il computer.

L'attività di progettazione dell'interfaccia utente di un qualsiasi sistema informatico e software che dialoga con l'utente attraverso uno schermo, viene intesa come Interface Design. In realtà l'interfaccia uomo/macchina (nell'accezione più comune la "macchina" è il computer), non è composta solo dall'interfaccia grafica che si visualizza virtualmente su uno schermo, ma comprende anche il sistema delle interfacce fisiche con cui l'utente interagisce, come tastiere, mouse, pulsanti, joystick, manopole, pannelli di controllo, segnalatori, indicatori e altri controlli fisici. I progettisti nel concepirle e realizzarle affrontano una miriade di piccole attività diversificate toccando tematiche che spaziano dal mondo dell'ingegneria, a quello dell'architettura e della progettazione pura fino alla comunicazione pura. I progettisti si trovano giornalmente di fronte a temi quali l'usabilità, la comunicazione, l'accessibilità e la tecnica di realizzazione ricercando costantemente il giusto punto di equilibrio.

Il processo che porta al progetto delle interfacce segue un andamento spiraliforme basato su quattro fasi (riportato in Figura 1):

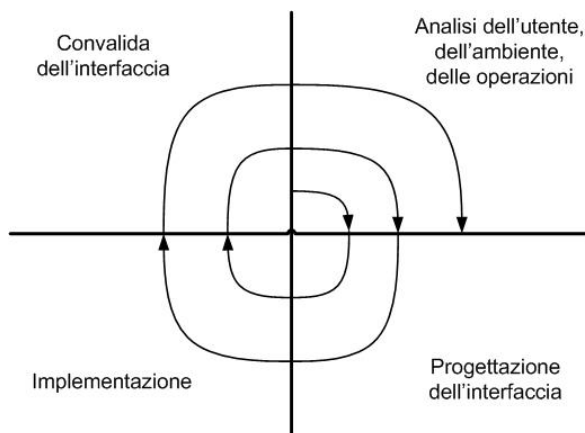


Figura 1 - Processo di progettazione dell'interfaccia

In queste quattro fasi non vanno mai tralasciati quelli che sono i principi di accessibilità e usabilità delle interfacce di un servizio web o di una applicazione informatica.

Usabilità e accessibilità sono fattori "ortogonali" che concorrono in modo indipendente a determinare la qualità dell'interfaccia. Tuttavia i processi di sviluppo tradizionali (raccolta requisiti, analisi funzionale, design, sviluppo, test, esercizio) sono inadeguati per gestire con efficienza i requisiti di usabilità ed accessibilità. Pertanto si parla di processo iterativo basato sulla produzione di prototipi. Nel dettaglio tale processo di sviluppo Human Centered si presenta come quello di Figura 2:

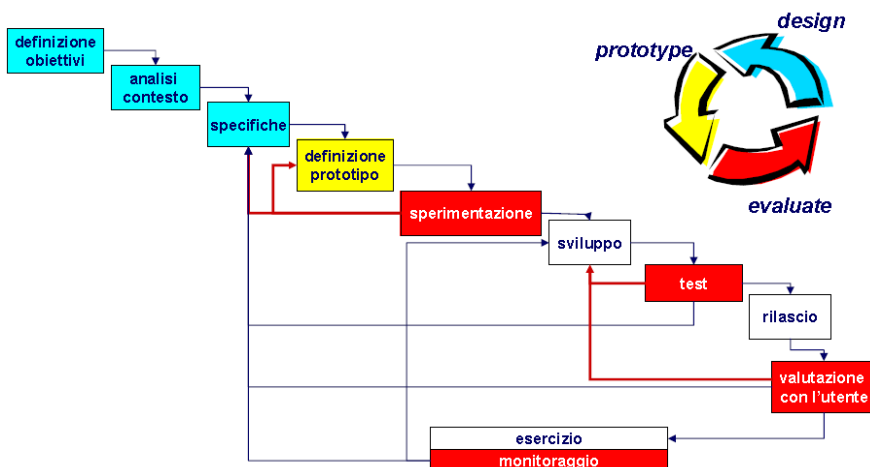


Figura 2 - Processo di sviluppo Human Centered

Con questo processo la qualità dell'interfaccia utente viene gestita e verificata in tutte le fasi del processo di sviluppo.

La ricchezza delle tecnologie informatiche moderne consente molti usi dei sistemi interattivi. Diventa spesso necessario che le interfacce utente si sappiano adattare al contesto di uso, che si può considerare da tre punti di vista: quelli relativi all'utente, al dispositivo e all'ambiente circostante. Per quanto riguarda l'utente, aspetti importanti sono gli obiettivi e i relativi compiti, le preferenze e il livello di conoscenza del dominio applicativo e delle modalità di interazione.

Nel dispositivo usato per l'interazione, è importante considerare le modalità supportate, l'ampiezza e la risoluzione dello schermo, le capacità e velocità di connessione con altri dispositivi, ecc. Infine, l'ambiente ha vari aspetti che possono influenzare le modalità di interazione come il livello di rumore e di luce corrente, gli oggetti che sono disponibili. Le interfacce utente si trovano, quindi, a doversi adattare a questi aspetti per avere una maggiore usabilità. Vi sono due tipologie di adattamento: **l'adattabilità**, ovvero la capacità di modificare aspetti su richiesta esplicita dell'utente in accordo a opzioni predefinite, **l'adattività**, ovvero la capacità del sistema di modificare aspetti dinamicamente senza richiesta esplicita dell'utente.

Mentre l'adattabilità consente essenzialmente di scegliere le modalità di interazione con un'applicazione tra un insieme predefinito, l'adattività implica sistemi che le cambiano dinamicamente rispetto al contesto d'uso. Questo da una parte implica una maggiore flessibilità, ma dall'altra implica che nuove problematiche di usabilità si possono generare se questi cambiamenti avvengono in modo non facilmente comprensibili per l'utente.

Una delle principali problematiche che ha impatto sulle interfacce utente, è la continua immissione sul mercato di nuove tipologie di dispositivi interattivi; interagire con essi diventa sempre più un'esperienza multi-dispositivo. Quando un sistema supporta modalità multiple (esempio: l'interazione vocale e grafica), le tecniche implementative devono tener conto dei diversi modi di combinare le modalità: complementare (entrambe le modalità sono usate in modo sinergico), assegnamento (una specifica modalità deve essere utilizzata per realizzare un certo scopo), ridondanza (più modalità sono usate per ottenere lo stesso scopo).

Nell'ambito della generazione delle interfacce utente non è sufficiente fare considerazioni solo sulla logica di presentazione. Questa stessa va opportunamente correlata alla logica di business in modo da determinarne il punto di contatto e quindi di sincronizzazione. Questo punto di sincronizzazione può essere individuato nelle entity: oggetti digitali, o risorse, dotati di rappresentazione semantica.

Il sistema SAPI sarà caratterizzato da entity composite (EC) ed atomiche (EA). Le prime individuano un concetto astratto e sono costituite da altre entity composite e/o atomiche; le seconde, invece, sono i contenuti veri e propri. Il numero di livelli non può essere definito a priori ma è funzione della complessità del contenuto che si deve presentare. Aver dotato le entity di descrizione semantica comporta che la fase di creazione si scinda in due ulteriori step: quella di progettazione della entity e quella di traduzione della semantica in un opportuno linguaggio di rappresentazione, nel caso specifico OWL.

Gli approcci che si possono seguire sono i seguenti.

- Bottom-up: a valle della raccolta di tutte le EA, si individuano EC di livello sempre più alto, e quindi più ricche.
- Top-down: sempre a valle della raccolta di EA, si individuano EC di livello sempre più basso. Quindi prima si raggruppano le EA in macroentity di primo livello e poi si procede per successivi raggruppamenti.

I due approcci hanno in comune la fase di raccolta delle EA che vanno opportunamente progettate. Le EA individuate vengono inserite in quella che viene detta SAPI Atomic Entity List (**SAEL**), che altro non è che una tabella formata da 2 colonne:

- *ID\_Entity*, identificativo univoco per le entity;
- *Entity\_Handler*, riferimento alla posizione fisica dove risiede l'implementazione della entity.

Tale tabella conterrà tutte le entity atomiche del sistema SAPI.

Progettare una EA significa:

1. individuare il contenuto che deve rappresentare;

Nell'ambito di un servizio, per poter individuare i contenuti che vanno presentati all'utente è necessario conoscerne la logica di business; in ambito SAPI si può avere una descrizione di alto livello sfruttando il modello funzionale di Input/Output. Pertanto una parte dei contenuti viene individuata sfruttando tale modello. Altri contenuti vengono invece individuati facendo riferimento alla logica di presentazione. Più precisamente bisogna cominciare ad entrare nelle logiche realizzative di essa e avere quindi un'idea di cosa e come lo si vuole presentare all'utente.

Pertanto, una volta individuato il servizio per cui si vogliono definire le EA, l'individuazione dei contenuti si articola negli step di seguito illustrati.

1. Selezione della activity per cui si necessita di individuare i contenuti.
2. Individuazione delle variabili che necessitano di una EA, a partire dal Modello Input/Output della activity atomica selezionata.
3. Verifica, nel SAEL, dell'eventuale esistenza di una EA che si adatta al caso.
4. Creazione della EA aggiungendo un riferimento nel SAEL (Solo se lo step 3 dà esito negativo).
5. Creazione della Business Entity Table (**BET**) strutturata con i seguenti campi:
  - *Nome Variabile*: nome generico associato alla entity che si è definita.
  - *ID Activity*: valore ricavato dall'ID del Modello Funzionale dell'activity in questione.
  - *SAEL Pointer*: puntatore al SAEL che permette di riferire l'EA nella sua totalità (l'EA nel SAEL è dotata anche di descrizione semantica).
6. Individuazione delle altre EA a partire dal Modello Funzionale dell'activity.
7. Verifica, sempre nel SAEL, dell'eventuale esistenza di EA che si adattano al caso.
8. Creazione delle EA aggiungendo i riferimenti nel SAEL (Solo se lo step 7 dà esito negativo).
9. Creazione della Presentation Entity Table (**PET**) strutturata con i seguenti campi:
  - *Nome Entity*: valore ricavato dal modello funzionale.
  - *ID Activity*: valore ricavato dall'ID del Modello Funzionale dell'activity in questione.
  - *SAEL Pointer*: puntatore al SAEL che permette di riferire l'EA nella sua totalità e di identificarla univocamente.

2. aggiungervi la descrizione semantica e quindi:

- 2.1. definire le informazioni semantiche, relative al contenuto stesso, a come si rapporta con entità simili e allo stile. Il tutto conforme ad uno standard (MPEG7 o Dublin Core)
- 2.2. definire le regole di adattamento, informazioni relative alle modalità con cui il contenuto può essere interpretato e gestito da un attore esterno. Tali modalità saranno molteplici in quanto ci si rivolge ad attori diversi a seconda del contesto di fruizione.

Tra le entity composte ne esisterà una particolare, Entity Composita del Servizio (ECS), che costituirà il contenitore di tutte le entity, composite ed atomiche, che serviranno perché si possa espletare il servizio cui si riferisce la ECS. Si tratta, pertanto, di una entity logica che viene composta all'occorrenza e, quindi, rappresenta il legame tra la logica di business di un particolare servizio e la relativa logica di presentazione. Fisicamente non esiste; è solo una entità logica che gerarchicamente raggruppa le entità fisiche, ossia le EA. Per poter definire una ECS è necessario individuare la gerarchia delle EC che la compongono e quindi le EA. Raccogliere tutte le EA che comporranno la ECS non è cosa semplice e veloce bensì alquanto complessa e meticolosa. A tal proposito è possibile utilizzare una tabella di ausilio, Service Atomic Entity Table (**SAET**), strutturata con le seguenti colonne:

- *Id Activity*: identificativo dell'activity ricavato direttamente dalle BET e PET.
- *Nome*: nome della variabile/entity definita nella activity e per la quale si è individuata una EA. Anch'esso viene ricavato direttamente dalle PET e BET.
- *SAEL Pointer*: puntatore al SAEL che permette di riferire l'EA e viene ricavato ancora dalle PET e BET.

Per la popolazione della SAET bisogna far riferimento alle PET e BET. Più precisamente per ogni activity bisogna:

1. riportare nella SAET tutte le entity elencate nella BET;
2. selezionare, tra tutte le entity esistenti nella PET, quelle che si ritengono più opportune, ai fini della presentazione e riportarle nella SAET. In questa fase la selezione è a discrezione del progettista.

Si passa ora ad analizzare nel dettaglio i due possibili approcci per poter individuare le entity composte ed atomiche.

Nella **progettazione bottom-up** si specificano nel dettaglio le parti elementari del sistema che coincidono nel nostro caso con le EA. Queste parti elementari vengono poi connesse tra loro in modo da formare componenti più grandi (le EC) che vengono a loro volta interconnessi fino a comporre la ECS. L'approccio bottom-up, pertanto, si articola nei seguenti step.

1. Comporre le EC sulla base della descrizione semantica delle EA presenti nella SAET.
2. Inserire il riferimento della EC appena individuata nel SAPI Composed Entity List (**SCEL**). Nel dettaglio i campi del SCEL assumeranno il seguente significato:
  - *ID\_Entity*, identificativo univoco per le entity composite;
  - *Nome\_Entity*, nome simbolico associato alle entity composite;
  - *Entity Reference*, puntatori al SAEL e al SCEL che permettono di riferire le EA e le EC che comporranno il SCEL. Infatti, nel caso in cui occorre riferire una EA, tale campo conterrà l'Id Entity del SAEL; se si tratta invece di una EC, conterrà l'Id Entity dello SCEL stesso.
  - *Service ID*, identificativo univoco del servizio cui appartiene la EC.
3. Aggiungere le varie sezioni della descrizione semantica di una:
  - 3.1. specificare tra i metadati della EC in questione le relazioni che occorrono tra le varie entity componenti;
  - 3.2. importare lo schema dei dati espresso nel documento di progetto della business logic con particolare attenzione ai tipi dei dati e ai constraints;

- 3.3. definire UIM (User Interaction Model) e UPM (User Presentation Model) con relative informazioni di stile;
- 3.4. definire NM (Navigation Model) una volta individuate le EC di primo livello. In pratica questo step viene temporaneamente evitato per poi essere applicato a tutte le EC individuate.

Questi tre step vanno seguiti a partire dalla definizione delle EC di livello più basso e ripetuti ricorsivamente fino alla individuazione della ECS.

Nella **progettazione top-down** a partire dalla ECS, si individuano man mano altre EC di livello inferiore. Ogni parte del sistema è successivamente rifinita aggiungendo maggiori dettagli. Ogni nuova parte così ottenuta (una nuova EC) può quindi essere nuovamente rifinita, specificando ulteriori dettagli finché la specifica completa è sufficientemente dettagliata da validare il modello.

L'approccio top-down, pertanto, si articola nei seguenti step che così come per l'approccio bottom-up vanno eseguiti ricorsivamente fino all'individuazione delle EC di più basso livello.

1. Comporre le EC di livello più alto sulla base della descrizione semantica delle EA presenti nella SAET. Quindi si individuano prima le macro-entity di livello superiore (EC di livello 1) per poi definire quelle di livello inferiore (livello n).
2. Inserire il riferimento della EC appena individuata nel SAPI Composed Entity List (SCEL).
3. Aggiungere le varie sezioni della descrizione semantica di una EC:
  - 3.1. specificare tra i metadati della EC in questione le relazioni che occorrono tra le varie entity componenti;
  - 3.2. importare lo schema dei dati espresso nel documento di progetto della business logic con particolare attenzione ai tipi dei dati e ai constraints;
  - 3.3. definire UIM e UPM con relative informazioni di stile;
  - 3.4. definire NM una volta individuate le EC di primo livello.

La definizione della ECS costituisce il passo propedeutico per eseguire i tre passi sopra riportati. In particolare nella definizione della ECS occorre definire, in linea di massima, quali EC faranno parte di essa. Ovviamente questa inclusione è logica e quindi richiede un notevole sforzo nel riuscire ad individuare, attraverso la logica di business, le EC da prendere in considerazione. Per raffinamenti successivi tali EC verranno ulteriormente dettagliate, individuando altre EC di livello inferiore, fino a considerare le EA che ne faranno parte, selezionate opportunamente dalla SAET.

Dopo aver illustrato le modalità con cui occorre individuare i contenuti si passa ora a mostrare a quella che viene detta logica di presentazione, attraverso la quale i contenuti vengono mostrati all'utente. L'elemento cardine della logica di presentazione è l'AIU (Abstract Interaction Unit), che non costituisce una pagina web ma un insieme di elementi funzionali attraverso i quali l'utente interagisce con il sistema. Essa, pertanto, può essere intesa come l'istanza di una EC. Una pagina web, quindi, sarà una opportuna composizione di AIU. Si parla di composizione, in quanto il sistema prevede AIU atomiche, che contengono solo widget (istanze di EA), e AIU composite che contengono una o più AIU (atomiche e/o composite). A corredo delle AIU vengono introdotte informazioni aggiuntive circa le modalità con cui l'utente potrà o dovrà interagire con i componenti dell'AIU. Tali informazioni costituiscono le indicazioni da utilizzare all'atto dell'implementazione relativamente ai controlli da effettuare per la fruizione del servizio. La progettazione delle AIU non costituirà un vincolo

definitivo per il sistema in quanto l'Evolver deve provvedere all'ottimizzazione di questa per ogni profilo utente.

Il sistema SAPI, pertanto, sarà comunque dotato di una versione di default per le AIU che viene presentata qualora l'utente che accede al sistema non sia ancora clusterizzato.

La Figura 3 sotto riportata illustra il processo di una AIU ottimizzata. Da essa si evince che per la generazione di una AIU l'evolver acquisisce UIM e UPM dal Layer dei Dati e procede all'ottimizzazione sulla base del profilo utente. Una volta generata la nuova istanza di UPM ottimo per lo specifico profilo si procede alla traduzione in OWL e quindi all'aggiornamento del Layer dei Dati. Il Deployer, integrando queste informazioni (OWL) con la logica di presentazione e i contenuti procede alla composizione dell'AIU che verrà presentata all'utente finale.

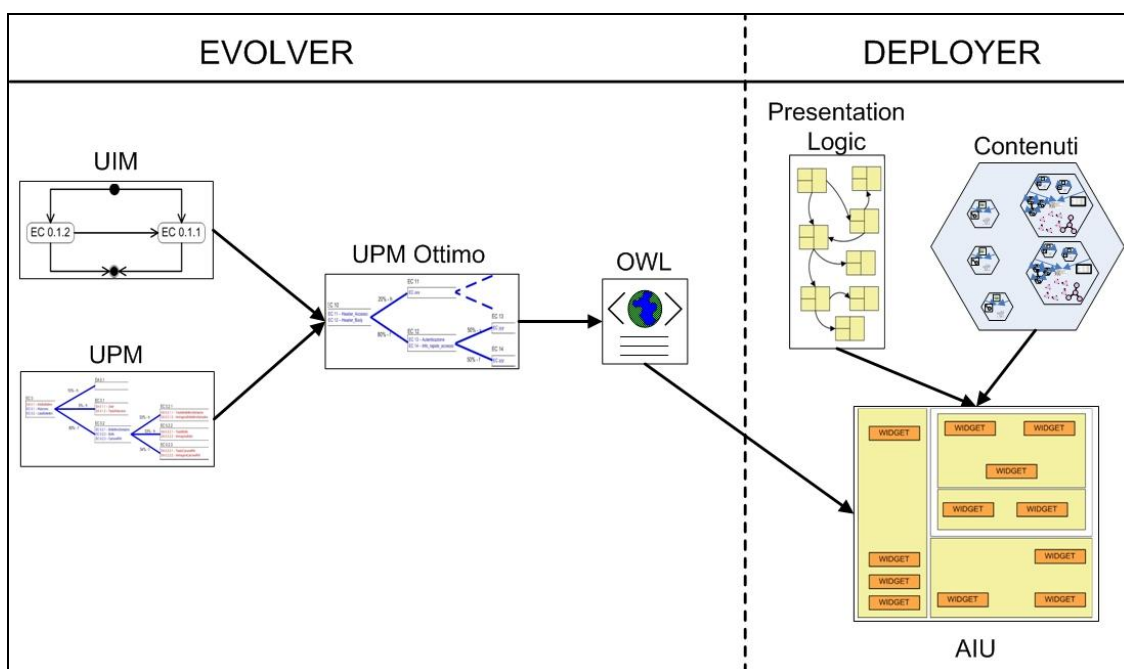


Figura 3: Processo di generazione di una AIU

Dalla figura, dunque, si evince che il processo di generazione di una AIU avviene a run-time ed è indipendente dal profilo cui viene associato l'utente che accede al sistema. Infatti, ciò che varia in base al profilo è la presentazione vera e propria, ma l'AIU è generata sempre allo stesso modo.

La definizione dell'AIU Schema (layout della pagina) risulta praticamente automatica una volta progettate le entity. Queste, infatti, contengono quasi tutte le informazioni relative a cosa deve essere presentato all'utente e alle modalità. Da ciò la necessità di riferirsi ai modelli della entity. In special modo allo UPM che contiene informazioni relative al layout delle EC. Tali informazioni sono quelle presenti sugli archi dello UPM. I nodi sorgenti e destinazione, infatti, verranno presi in considerazione in fase di composizione dell'AIU, ossia quando bisogna riempire l'AIU Schema con i contenuti. Le informazioni sugli archi dello UPM sono:

1. **PageSequence:** indica in quale pagina va presentata la entity in questione.
2. **OrdineVisita:** indica quale è l'ordine di visita dell'albero che rappresenta lo UPM.

3. **Taglio:** indica come la sezione dedicata alla entity sorgente va suddivisa. I valori che può assumere sono 'h' (orizzontale), 'v' (verticale), 'f' (finale, senza ulteriore suddivisione).
4. **Importanza:** è un valore percentuale che indica il peso della suddivisione della sezione dedicata a quella entity.

Tali informazioni vengono interpretate dall'UI Generator che le acquisisce dal Layer dei Dati e dà vita all'AIU Schema (Figura 4).

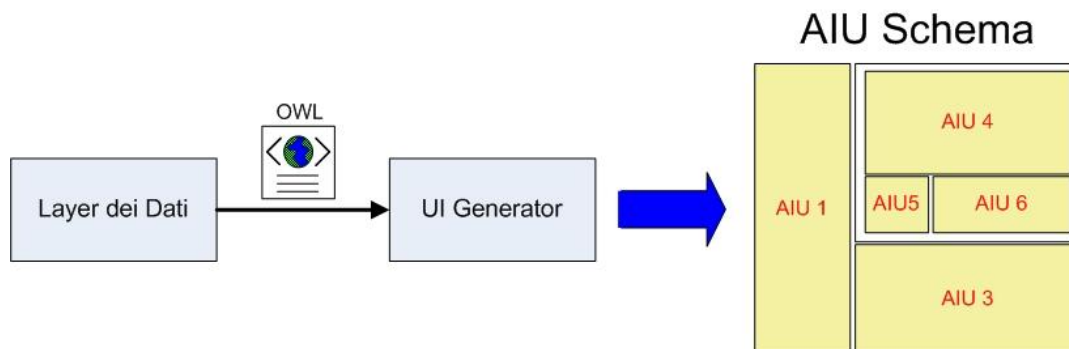


Figura 4: Generazione AIU Schema

Ovviamente l'OWL che riceve in ingresso l'UI Generator terrà conto del profilo utente. Chi permette la generazione di diverse AIU Schema per una stessa entity è l'Evolver che ha proceduto all'ottimizzazione dello UPM tenendo conto del profilo dell'utente e dell'UIM.

Da quanto esposto sinora è possibile dedurre che le AIU sono funzione delle entity che andranno a contenere e dello stereotipo utente cui sono destinate. Le AIU prevedono anche la logica di presentazione che tiene conto degli aspetti legati alla tecnologia. Si tratta, quindi, di aspetti a corredo della AIU che non prevedono ottimizzazione e che, a differenza dei modelli definiti per la entity, non sono destinati al sistema quanto piuttosto al developer che implementa le soluzioni anche in funzione di essi. Inoltre tale logica porta in conto informazioni da fornire e/o scambiare con i sensori cognitivi che devono rilevare le modalità di interazione (numero di click, tempo atteso, eventi *back-forward*, ...). Quindi è necessario che l'AIU, in quanto definita a tempo di progetto, contenga informazioni strettamente legate a come l'utente dovrebbe comportarsi.

La parte finale del documento presenta gli aspetti tecnologici (linguaggi e architetture) da considerare per l'implementazione dell'AIU.

Tra le tecnologie utilizzabili per creare le AIU, non si può che partire da quelle definite dalla W3C. In particolare si considerano tecnologie quali DIAL (Device Independent Authoring Language) e XForms come possibili linguaggi utilizzabili per costruire le AIU o parte di esse.

Si passa poi ad analizzare le tecnologie RIA, ossia applicazioni web che hanno le caratteristiche e le funzionalità delle tradizionali applicazioni desktop (cioè residenti sul computer). Le RIA vengono eseguite all'interno del web browser che funge come un vero e proprio contenitore. Nelle RIA tipicamente è trasferita a livello client la parte dell'applicazione che processa i dati e fornisce una pronta risposta all'interfaccia utente, mentre la gran parte dei dati e dell'applicazione rimane sul server. Un approccio di questo tipo permette da un lato di ottenere un feedback molto più rapido ed un'interazione uomo-macchina più friendly,

dall'altro però appesantisce e complica il client in quanto viene introdotto un nuovo strato di software, detto client-engine, tra lo user ed il server. Il *client-engine* si deve occupare:

1. rendering della UI, cioè decidere cosa (*Content*) e come (*Layout e Look&Feel*) visualizzare la UI,
2. comunicazione con il server.

La Figura 5 riporta una tipica architettura RIA, che risulta indipendente dalla tecnologia implementativa utilizzata.

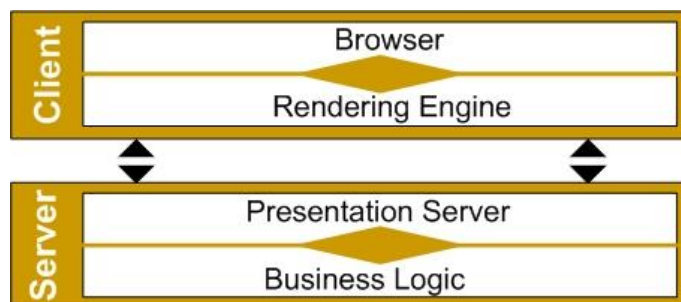


Figura 5: Tipica architettura RIA

Le tecnologie RIA che attualmente sono presenti sul mercato sono le seguenti:

1. Tecnologie Microsoft (Silverlight):

Silverlight fonda le sue prerogative sulla ben nota piattaforma .NET, favorendo in questo modo chi programma già con i linguaggi Microsoft. Per usare applicazioni Silverlight è necessario un plug-in, ovvero un componente aggiuntivo che permette al browser di visualizzare contenuti non tradizionali. Silverlight è stato sviluppato in due versioni (1.0 e 2.0). Le interfacce utente vengono rappresentate con **XAML**: un linguaggio di markup derivato da XML, col quale si possono rappresentare oltre a testo e bottoni, anche diversi tipi di elementi grafici, trasparenze e le animazioni degli oggetti.

2. Tecnologie Open Source (OpenLaszlo)

OpenLaszlo è una piattaforma open source per lo sviluppo di web applications con un'interfaccia utente utilizzabile sul World Wide Web. OpenLaszlo è rilasciata sotto la licenza Open Source Initiative-certified Common Public License. OpenLaszlo è basato sul linguaggio **LZX** e su un OpenLaszlo Server.

3. Tecnologie Adobe (Flash e Flex)

Flex è caratterizzato per lo sviluppo di interfacce utente usando un linguaggio XML chiamato **MXML**. Il linguaggio Flex e la sua strutturazione in sorgenti MXML per la GUI ed ActionScript per la Business Logic sono studiati per distinguere la logica della programmazione dal design implementando di fatto il Design pattern MVC.

Un'altra importante scelta di progetto è decidere dove l'adattamento deve essere svolto. A questo proposito le possibilità sono:

- Sul server.  
Il server, fornitore dei contenuti, ha anche la responsabilità di analizzare il profilo utente per sintetizzare la strategia di adattamento più appropriata.
- Su un intermediario, proxy.

Il Client è connesso al Server tramite un intermediario detto Proxy che intercetta le richieste del terminale utente, recupera dal Server il contenuto richiesto e lo adatta secondo le strategie e politiche che esso stesso decide di usare in base al contesto dell'utente.

- Sul client.

Il processo di adattamento viene effettuato dal terminale utente che riceve dal Server una risposta che include una lista di rappresentazioni del contenuto richiesto, dalla quale il terminale seleziona quella più appropriata alle proprie caratteristiche.