

QUANDO L'UTENTE GUIDA L'INNOVAZIONE IL WEB MASHUP

Una pratica sempre più emergente nello scenario del Web 2.0 è lo sviluppo di mashup, applicazioni Web basate sulla composizione di contenuti e servizi aperti. In questo articolo illustreremo come, tramite strumenti e processi di sviluppo leggeri, anche gli utenti meno esperti possano assumere il ruolo di sviluppatori di mashup. Evidenzieremo la capacità dei mashup di far leva sul potenziale di innovazione degli utenti del Web e la necessità in questo contesto di metodi di sviluppo rinnovati. Traceremo quindi le maggiori problematiche che dovranno essere affrontate affinché i nuovi metodi trovino applicabilità.

1. INTRODUZIONE

La tendenza corrente nello sviluppo delle moderne applicazioni Web, e in particolare delle applicazioni del Web 2.0, punta chiaramente verso il coinvolgimento sempre più marcato dell'utente. Le cosiddette applicazioni sociali sono la prova del valore, inizialmente inaspettato, dell'integrazione degli utenti finali nel processo di creazione dei contenuti. Un'altra pratica emersa recentemente è lo sviluppo di *Web mashup*, applicazioni Web ottenute dalla combinazione di contenuti e servizi disponibili sul Web sotto forma di API (*Application Programming Interface*, cioè interfacce software di programmazione) "aperte" o, più in generale, di servizi riusabili. L'esempio più affermato di mashup è www.housingmaps.com, nato dall'integrazione tra gli annunci di affitto pubblicati su www.craigslist.com e le mappe di Google. Il risultato è un'applicazione Web innovativa, che semplifica la localizzazione degli appartamenti offerti in affitto.

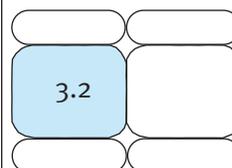
Il primo e fondamentale passo nello sviluppo dei mashup è la produzione di *servizi aperti*, pubblicati sul Web e pertanto facilmente ac-

cessibili e riutilizzabili. L'uso di servizi aperti è una caratteristica unica, che distingue lo sviluppo dei mashup da altri paradigmi di sviluppo basati su componenti (riquadro a p. 30 per un approfondimento sui mashup e sulle tecnologie abilitanti). Tali servizi sono eterogenei, e possono essere API remote, servizi basati sullo scambio di messaggi (per esempio, servizi Web), API basate sull'integrazione programmatica di codice (come accade per le API di Google Maps e di Twitter), o anche feed RSS/Atom (per esempio, notizie sui titoli di borsa), o contenuti estratti dai siti Web più disparati (per esempio, i prezzi di certi prodotti). Tali servizi possono inoltre essere dotati di un'interfaccia utente propria (si parla in tal caso di servizi UI), che poi diventa parte integrante dell'interfaccia del mashup finale, possono offrire supporto alla computazione di certe funzionalità, o possono agire come semplici sorgenti di dati.

I mashup più popolari integrano API pubbliche che permettono a utenti e sviluppatori generici di invocare servizi accessibili tramite Web. Tuttavia, una prospettiva promettente è



Florian Daniel
Maristella Matera



Web mashup

I mashup sono applicazioni web – nella maggior parte dei casi molto semplici, per esempio composti da una sola pagina Web – sviluppate a partire da componenti software accessibili tramite il Web, come per esempio contenuti e dati, servizi che forniscono logica applicativa oppure elementi di presentazione (pezzi di interfaccia utente, solitamente in formato HTML) [4]. Casi tipici possono essere trovati facilmente, per esempio, sul sito programmableweb.com diventato nel corso degli ultimi anni uno dei riferimenti principali per sviluppatori di mashup, specialmente per quanto riguarda la ricerca di componenti base. I mashup sono quindi una pratica di composizione e di riuso di componenti pubblicati sul Web, che aggiunge valore agli stessi in quanto trova forme di utilizzo nuove e creative che spesso vanno ben oltre lo scopo originario dei singoli componenti. Basti pensare che i primi mashup si basavano soprattutto sul riuso di contenuti estratti da siti Web convenzionali senza che il proprietario dei contenuti ne fosse a conoscenza.

La composizione di applicazioni o di processi a partire da servizi esistenti non è una disciplina nuova. Sono ormai anni che la comunità scientifica studia metodi e linguaggi di composizione per i cosiddetti *Web service* con risultati molto buoni (pensiamo per esempio a BPEL o ai vari approcci di *Enterprise Application Integration*), e il campo dell'integrazione di dati è ormai un'area consolidata, nonostante le nuove sfide poste da dati sempre meno strutturati. L'aspetto innovativo introdotto dai mashup nel mondo della composizione è che, diversamente dagli approcci d'integrazione a livello di dati o di logica applicativa, integrano anche elementi di presentazione, un aspetto facilitato dall'uso di tecnologie web standardizzate.

I componenti usati nello sviluppo di mashup sono quindi di tre tipi:

□ Servizi di dati come i feed RSS (*Really Simple Syndication*) o Atom, contenuti formattati in JSON (*JavaScript Object Notation*) o XML oppure semplici file di testo. Per esempio, quasi tutti i giornali pubblicano ormai i titoli delle loro notizie tramite feed RSS che possono essere letti da un cosiddetto RSS reader e permettono all'utente di saltare facilmente dettaglio delle varie notizie.

□ Servizi Web o API (*Interfacce Applicative Programmabili*) accessibili tramite il Web come i servizi SOAP (*Simple Object Access Protocol*) o REST (*REpresentational State Transfer*). Questi servizi tipicamente non forniscono semplici dati, ma permettono il riuso di logica applicativa come, per esempio, il calcolo del nome di una città a partire dalle sue coordinate GPS.

□ Componenti UI (cioè dotati di interfaccia utente) come pezzi di codice HTML o interfacce programmabili in JavaScript (per esempio, i cosiddetti widget – <http://www.w3.org/TR/widgets-apis/>). Il tipico esempio di componente UI è Google Maps, che fornisce non solo dati in forma di carte geografiche ma anche un'interfaccia utente facilmente integrabile in una pagina Web che permette all'utente di navigare le mappe. Comunque, anche l'estrazione di contenuti da pagine Web tradizionali è ancora una prassi molto diffusa, specialmente in assenza di servizi equivalenti già disponibili e pronti per l'uso.

Data questa eterogeneità di componenti – e quindi di tecnologie e di logiche applicative diverse da componente a componente – lo sviluppo di un mashup è tutt'altro che facile e spesso ancora una prerogativa di sviluppatori esperti. Sono loro che conoscono i linguaggi di programmazione, i protocolli di comunicazione, e gli standard per implementare la logica di integrazione necessaria per mettere in comunicazione i vari componenti e l'impaginazione grafica dei componenti all'interno del layout del mashup.

La situazione è ancora più complessa qualora i componenti necessari non fossero già disponibili online. In questo caso è lo sviluppatore che deve prima implementare il componente per poi poterlo comporre. Nella maggior parte delle volte questo vuol dire estrarre dati da una o più pagine Web, una pratica che tipicamente richiede l'uso di espressioni regolari complesse e sforzi significativi di programmazione. Per alleviare tali sforzi, di recente sono emersi strumenti online, che sono in grado di assistere lo sviluppatore nell'estrazione di dati da pagine Web tramite interfacce grafiche e interattive e che permettono di selezionare i contenuti desiderati e di ottenere automaticamente un componente riusabile. Esempi di questi strumenti sono Dapper o Openkapow, tutti e due accessibili online e gratuiti.

Comunque, come anche proposto in questo articolo, la visione della comunità scientifica e dell'industria del software è quella di mettere a disposizione dell'utente Web medio degli strumenti di sviluppo e ambienti di esecuzione che siano semplici ed intuitivi (per esempio basati su linguaggi di composizione grafici e paradigmi *drag-and-drop*) e che permettano all'utente stesso di sviluppare applicazioni personali a partire da componenti esistenti in maniera assistita e senza avere bisogno dell'aiuto da parte di programmatori esperti. Strumenti come Yahoo! Pipes, Intel Mash Maker, IBM Lotus Mashups o JackBe Presto rappresentano un primo passo in quella direzione, ma ancora non possiamo parlare di utenti veramente autosufficienti. Spesso tali strumenti forniscono semplificazioni tecnologiche rilevanti, ma quello che manca all'utente Web medio sono le nozioni base dello sviluppo di software. In questo contesto, il design di meccanismi che assistano l'utente tramite raccomandazioni di sviluppo interattive è una delle sfide di ricerca del futuro.

data dai *mashup aziendali* [7], un'estensione dei mashup verso le intranet, per permettere ai membri di un'azienda di utilizzare servizi interni per l'accesso alle risorse informative dell'azienda stessa e “mescolarli” in modi innovativi, per esempio per automatizzare una procedura burocratica ricorrente.

Entrambi i tipi di servizi, sia le API pubbliche, sia i servizi interni all'azienda, possono incoraggiare gli sviluppatori e gli utenti a scoprire e

promuovere usi innovativi. Se coadiuvati da strumenti di sviluppo e metodologie adeguate, anche gli utenti Web meno esperti sul piano delle tecnologie potrebbero far uso di tali servizi per creare applicazioni nuove, trasformandosi così da ricettori passivi d'innovazione ad attori direttamente coinvolti nella creazione di innovazione. L'aggregazione di questo processo su tutti gli utenti che in parallelo conducono esperimenti con gli stessi servizi accelera

enormemente l'innovazione, e copre uno spazio di progettazione più ampio rispetto a quello che i fornitori originari riuscirebbero a concepire se non aprissero i loro servizi vero parti esterne. Lo sforzo che molti dei maggiori produttori dell'economia di internet odierna (per esempio, IBM, Intel, Yahoo!, SAP) stanno investendo nella ricerca sui mashup, testimonia che qualcosa di nuovo sta accadendo, che va ben oltre le soluzioni basate sulla programmazione spinta a cui attualmente ricorrono i programmatori di mashup esperti.

In questo articolo, esploreremo il mondo dei mashup e il loro potenziale come strumenti tramite cui gli utenti del Web, non necessariamente esperti di tecnologie, possono creare innovazione. Spiegheremo perché gli utenti del Web possono trovare interesse nello sviluppare in prima persona le loro applicazioni, e quali altre figure possono trarne benefici. Discuteremo quindi il processo di sviluppo dei mashup e gli strumenti e i metodi che possono supportare anche gli utenti più inesperti nell'innovare attraverso i mashup. Questa discussione ci permetterà di dare uno sguardo critico riguarda come i mashup sono sviluppati oggi e di individuare quindi cosa manca per realizzare i mashup del futuro.

2. INNOVAZIONE CREATA DAGLI UTENTI E STRUMENTI PER L'INNOVAZIONE

Cosa spinge gli utenti a sviluppare le proprie applicazioni? Esiste un fattore specifico che guida il fenomeno dei mashup: *la possibilità, offerta agli utenti, di innovare*, cioè il desiderio e la capacità degli utenti di sviluppare "prodotti" propri e di realizzare le proprie idee, esprimendo così le proprie capacità personali. Per comprendere meglio questo fattore trainante, consideriamo il fenomeno dilagante di Twitter, un servizio di microblogging attraverso cui amici e collaboratori possono rimanere in contatto e scambiarsi opinioni. Circa l'80% del traffico di Twitter parte attraverso la sua API pubblica; ciò vuol dire che il traffico è generato principalmente da applicazioni web che al loro interno integrano tale servizio. I fondatori di Twitter considerano l'API pubblica la loro più importante

scelta di innovazione¹: "Ci ha permesso in primo luogo di mantenere il servizio semplice e di creare uno strumento di facile utilizzo tramite cui gli sviluppatori possono costruire applicazioni nuove sulla base della nostra piattaforma e venir fuori con idee [come per esempio *Twitterific*] che sono decisamente migliori [grazie all'innovazione creata dagli utenti] delle nostre". È così che Twitter è diventata un'infrastruttura per applicazioni Web create dagli utenti del Web, secondo un paradigma che l'economista Thomas Hughes ha definito *invenzione conservativa* [4]: l'innovazione generata da terze parti contribuisce alla crescita di un sistema, mentre il sistema stesso continua a rimanere sotto il controllo del suo produttore originale.

In un ciclo tradizionale di progettazione-sviluppo-valutazione, il riscontro del cliente che commissiona l'applicazione diventa disponibile ai progettisti solo dopo il rilascio di un prototipo, quando ogni modifica può comportare costi considerevoli. In un approccio basato sull'innovazione guidata dagli utenti, un'azienda offre un *pacchetto di innovazione* attraverso cui i clienti stessi possono costruire il loro prodotti [12]. Questo pacchetto fornisce un'interfaccia limitata sulle potenzialità della piattaforma dell'azienda, così da assicurare che i nuovi prodotti siano costruiti in modo corretto e appropriato (per esempio senza violazioni d'uso). L'idea alla base è quindi che la sperimentazione iterativa, necessaria per sviluppare un nuovo prodotto, può essere portata avanti interamente dall'utente stesso del prodotto. Diversi utenti possono lavorare in parallelo alla soluzione di un problema, ognuno concentrandosi su una propria versione. Essi possono creare soluzioni ad hoc per i loro requisiti e possono velocemente verificarne la fattibilità tramite i loro esperimenti di sviluppo. Allo stesso tempo, l'azienda che fornisce il pacchetto di innovazione non deve sostenere il costo di produzioni senza successo. Inoltre, quando un esperimento produce valore nuovo e rilevante, l'azienda ha il diritto di integrare l'innovazione nel suo prodotto originario. Questo è quanto successo nel Web quando alcuni

¹ <http://readwritetalk.com/2007/09/05/biz-stone-co-founder-twitter>

sviluppatori integrarono in un mashup il servizio di Flickr con un servizio di mappe: Flickr ha successivamente incorporato una funzionalità di mappa sia nella sua piattaforma Web, sia nel suo servizio pubblico. Anche Google monitora costantemente l'uso delle sue API pubbliche (come per esempio Google Maps e Google Search) per raffinarle "appropriarsi" degli usi innovativi migliori [6].

L'apertura dei servizi affinché terze parti possano utilizzarli nei propri mashup è quindi una scelta strategica, che rivoluziona il modello di business che per anni ha caratterizzato il Web e le sue applicazioni. Piuttosto che continuare a essere ricettori passivi d'innovazione, gli utenti del Web diventano attori attivi nella creazione dell'innovazione. Il controllo sempre

più crescente sulla proprietà intellettuale della soluzione creata fornisce un incentivo per gli utenti affinché essi si assumano le responsabilità dello sviluppo del software. Questo significa anche che il valore creato dall'innovazione è ora condiviso con gli utenti. Il valore globale è con buona probabilità più grande, così che la condivisione con gli utenti rimanga comunque vantaggiosa.

In tale contesto, l'innovazione del software è divenuta sempre più sistemica [8]. Un'azienda dipende sempre più da fornitori esterni per la produzione di servizi che estendono un prodotto base. I fornitori esterni non sono sotto il controllo diretto dell'azienda. Quindi, per poter contare su di essi, l'azienda deve dimostrare un impegno credibile nell'innovazione sistemica. I fornitori di servizi aperti, per esempio, cercano di attrarre fornitori esterni offrendo libero accesso ai loro dati e riducendo il controllo sulla proprietà dei dati e dei formati di scambio usati dalle API. I progressi dei mashup sono quindi il risultato logico di una tendenza generale a incrementare la complessità dei sistemi e allo stesso tempo la loro specificità. La complessità è legata al numero di tecnologie e di componenti incorporati in un prodotto. La specificità si riferisce invece al fatto che ogni azienda coinvolta nello sviluppo ha la possibilità di concentrarsi sulle competenze in cui eccelle, aumentando così l'efficienza del processo di produzione.

3. LO SCENARIO DI SVILUPPO DEI MASHUP

Il modo in cui i mashup sono sviluppati dipende dal tipo di mashup. Attualmente, i mashup destinati ai "consumatori" (per esempio, tutti i numerosi mashup che integrano mappe) sono perlopiù il risultato di attività di programmazione da parte di sviluppatori esperti. I mashup "aziendali" evidenziano invece diversi scenari di sviluppo, che vedono anche il coinvolgimento di utenti meno esperti. Prenderemo quindi spunto da studi recenti sui mashup aziendali [10, 9] e cercheremo di evidenziare i contributi che diversi attori, con diversi livelli di esperienza tecnica, possono apportare al loro sviluppo.

La figura 1 illustra tre diversi scenari, che differiscono per l'eterogeneità dei servizi integrati nei mashup, la diversità dei bisogni de-

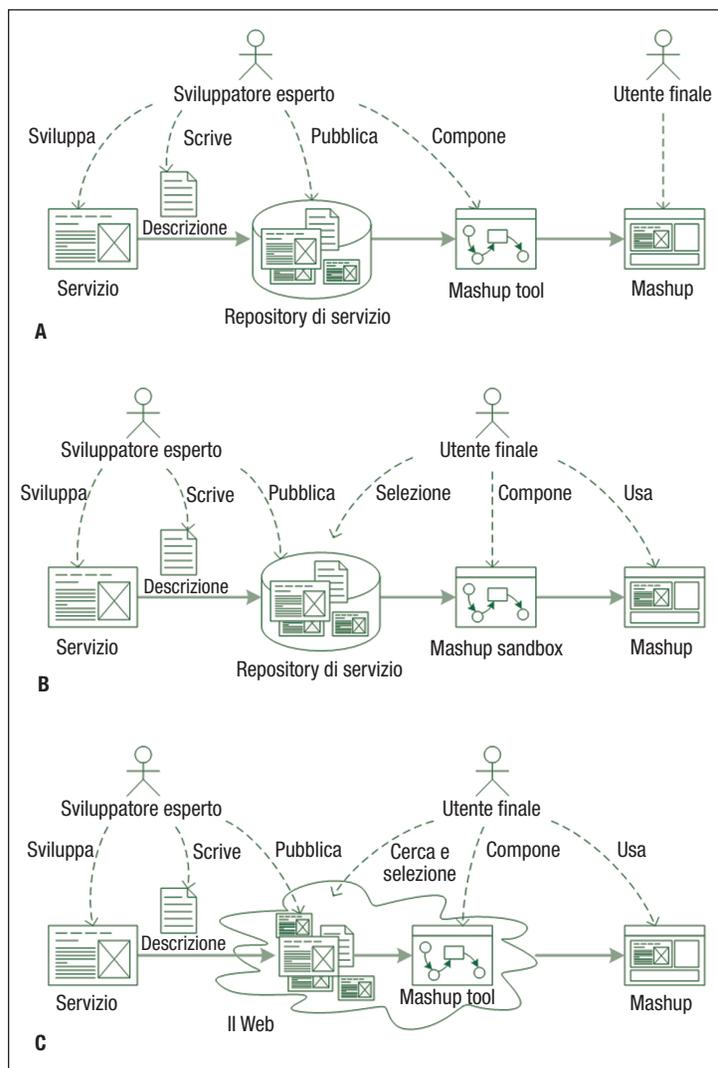


FIGURA 1
Scenari per l'uso dei mashup come tecnica di sviluppo software

gli utenti, e il grado di sofisticazione degli utenti e degli strumenti che supportano il loro lavoro:

A. I mashup possono essere creati da sviluppatori esperti (per esempio sviluppatori di un dipartimento IT o fornitori di servizio) in grado di rilasciare applicazioni in breve tempo. Gli utenti finali non sono coinvolti nella produzione dei mashup ma traggono in ogni caso benefici dai tempi di sviluppo ridotti.

B. Gli sviluppatori esperti creano servizi adatti a essere composti in un mashup e forniscono allo stesso tempo un ambiente ben delimitato (sandbox) dove gli utenti finali possono combinare i servizi. Il ruolo degli sviluppatori è quindi creare servizi, mentre i mashup sono costruiti dagli utenti, i quali sono maggiormente a conoscenza del dominio applicativo in cui i mashup saranno utilizzati.

C. Gli sviluppatori esperti rilasciano un ambiente che permette a ognuno di creare i propri mashup. Questo è analogo al modo in cui i fogli di calcolo sono usati oggi nelle organizzazioni: gli utenti finali, per esempio gli analisti finanziari di un'azienda, creano i propri fogli di calcolo senza dover necessariamente coinvolgere un programmatore esperto. I mashup sono spesso creati per un singolo scopo e per un unico utente e sono legati a una situazione specifica. Queste applicazioni sono, infatti, note come *situational applications* [1].

Quando i mashup sono sviluppati in modo centralizzato (scenario A), lo sviluppo è affidato ai soli sviluppatori esperti del dipartimento IT. Date le risorse limitate di un dipartimento IT, potranno essere sviluppate solo le applicazioni più richieste. Tuttavia, gli sviluppatori esperti hanno l'esperienza e le capacità necessarie a integrare servizi largamente eterogenei a un basso livello di astrazione, e possono quindi essenzialmente produrre applicazioni di ogni tipo.

Quando il dipartimento IT concentra le sue risorse sullo sviluppo di componenti e sul supporto necessario agli utenti per usare tali componenti (scenario B), esporrà certi tipi di dati e servizi in un formato facile da usare da utenti non sviluppatori. Rispetto ai programmatori esperti, questi utenti avranno una migliore conoscenza del dominio applicativo. Poiché gli sforzi per lo sviluppo dei mashup non sono totalmente a carico dello staff IT, è possibile sod-

disfare un maggior numero di richieste degli utenti. Tuttavia, i mashup che possono essere costruiti sono esclusivamente limitati ai componenti forniti dallo staff IT: la composizione del mashup consisterà in generale nella parametrizzazione dei componenti, poiché gli utenti non hanno la capacità di eseguire un'integrazione più "profonda" e articolata.

La produzione di uno strumento per lo sviluppo di mashup (scenario C) è sicuramente la soluzione più impegnativa da realizzare, ma allo stesso tempo la più proficua per i vari attori. Attraverso tale strumento, gli utenti possono integrare servizi e dati per creare i propri mashup. Rispetto ai mashup programmati "a mano", lo strumento limita le opzioni di integrazione al fine di garantire la correttezza della composizione, ma allo stesso tempo concede agli utenti la libertà di integrare qualsiasi tipo di componente disponibile nel Web. Tale strumento fa cioè in modo che gli utenti possano creare le proprie applicazioni, che possono essere di diversi tipi così da assecondare un'ampia diversità di requisiti [12].

Un'altra differenza tra i tre diversi scenari riguarda il grado di controllo sulla qualità dei mashup creati. Nello scenario A, lo sviluppo da parte dello staff IT è garanzia di qualità. Tuttavia, non tutti i mashup hanno requisiti stringenti in termini di sicurezza, prestazioni, o affidabilità, soprattutto se destinati a essere usati per uno scopo ben specifico, per il quale una soluzione complessa sviluppata da un dipartimento IT sarebbe troppo costosa. Nello scenario B, il dipartimento IT seleziona i componenti che possono essere integrati nei mashup degli utenti e fornisce inoltre una piattaforma per comporre ed eseguire i mashup. Gli utenti sono quindi abilitati a creare mashup per soddisfare bisogni specifici che difficilmente potrebbero essere previsti e assecondati dal dipartimento IT. I mashup sviluppati dagli utenti nello scenario C possono in seguito essere utilizzati come prototipi per rendere più solide le applicazioni sviluppate dal dipartimento IT, per esempio nel caso in cui dovesse emergere il bisogno di rendere disponibili quei mashup a diversi utenti all'interno dell'azienda o a clienti esterni.

Secondo von Hippel [13], i mashup rendono l'innovazione "democratica", permettendo agli utenti finali di soddisfare quei bisogni cui

un dipartimento IT centralizzato, o più in generale un fornitore di servizio, non è sempre in grado di rispondere. Il loro uso riduce i tempi necessari a implementare una certa funzionalità. Uno dei possibili usi dei mashup è, infatti, la prototipazione di soluzioni a problemi di utenti specifici, che possono in seguito essere generalizzate per soddisfare i bisogni di una comunità più ampia. Lo sviluppo dei mashup è quindi simile allo sviluppo *open source* (che ha ispirato la metafora di von Hippel) secondo due diverse sfaccettature:

- coloro che contribuiscono allo sviluppo di un progetto open source sono anche utenti del software prodotto;
- i progetti open source forniscono un meccanismo attraverso il quale, chi contribuisce attivamente al progetto passa dall'essere un utente passivo al fornire utili commenti, richieste di funzionalità aggiuntive e in alcuni casi pezzi di codice.

In modo simile, gli sviluppatori di mashup sono spesso anche gli utenti dei mashup (vedi scenari B e C nella Figura 1); non tutti gli utenti dei mashup rivestono necessariamente il ruolo di sviluppatori, ma possono comunque contribuire allo sviluppo fornendo commenti e richiedendo nuove funzionalità durante lo sviluppo dei prototipi.

4. PROCESSI DI SVILUPPO LEGGERI

A partire dagli scenari di sviluppo tipici dei mashup, è possibile fare qualche considerazione su come i processi di sviluppo per il Web debbano modificarsi per soddisfare le esigenze di questa nuova classe di applicazioni. È noto come il ciclo di vita delle applicazioni

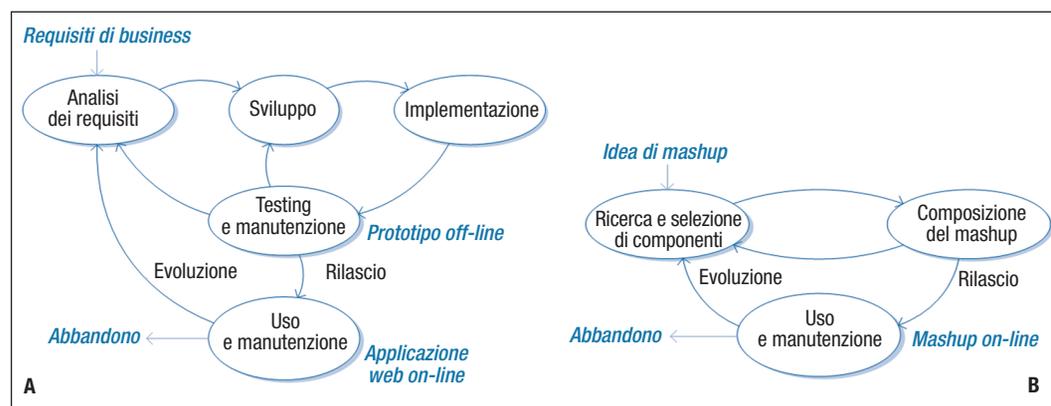
Web sia già molto più dinamico rispetto ad altre classi di applicazioni software:

1. lo sviluppo per il Web deve essere veloce: i prototipi e le applicazioni finali devono essere sviluppate nei "tempi di Internet", e cioè in giorni o settimane, invece che in mesi o anni;
2. la possibilità di registrare i dati di utilizzo di una grande mole di utenti facilita e snellisce l'attività di testing e le analisi di usabilità;
3. dopo il rilascio di un'applicazione, eventuali evoluzioni e raffinamenti possono essere realizzati mentre l'applicazione è ancora online; in altre parole, un'applicazione Web è potenzialmente soggetta a continue evoluzioni.

Lo sviluppo per il Web si articola quindi in due fasi principali: *lo sviluppo incrementale* della versione base dell'applicazione e *l'evoluzione incrementale* successiva al rilascio. Se pur dinamico e veloce, questo processo va ben oltre i requisiti di sviluppo dei mashup. Sposa bene le caratteristiche e le capacità dei programmatori esperti che sviluppano applicazioni Web tradizionali - si veda la figura 2 A - ma non incontra le aspettative e le capacità degli utenti che vogliono prender parte allo sviluppo; quindi, non promuove l'innovazione.

Il processo di sviluppo ideale per i mashup dovrebbe riflettere il potenziale di tali applicazioni: poter *comporre* un'applicazione partendo da *contenuti* e funzionalità che rispondono a bisogni personali, e poterla *eseguire* con facilità, senza doversi preoccupare di ciò che succede dietro la scena. Lo sviluppo iterativo, centrato sul rilascio di prototipi, è ancor più accentuato: il compositore di mashup combina servizi aperti ed esegue il risultato per controllare che tutto funzioni correttamente e soddisfi i suoi bisogni. In caso di risultati insoddisfacenti, il compositore stesso risolve i

FIGURA 2
A - Modello del ciclo di vita delle applicazioni web attuali, B - Modello del ciclo di vita dei mashup



problemi e quindi esegue immediatamente il nuovo mashup per provare l'efficacia delle modifiche apportate. Data la natura del mashup, strettamente legata alla specifica situazione di utilizzo, il ruolo del committente dell'applicazione deve essere reinterpretato: i requisiti iniziali, di fatto, corrispondono a quei bisogni "di breve termine" del compositore del mashup, dai quali ha origine l'idea di sviluppare il mashup. Inoltre, le fasi di *progettazione, implementazione, testing e valutazione* assumono un'importanza diversa, e dovrebbero essere in qualche modo semplificate (o addirittura nascoste) attraverso l'uso di strumenti adeguati.

Così come mostrato nella figura 2 B, queste considerazioni possono essere riassunte in un modello che promuova un *processo di sviluppo leggero*, caratterizzato da tre attività principali:

□ *scoperta e selezione*: partendo da un'idea iniziale di mashup, che generalmente riflette bisogni e preferenze personali, il compositore seleziona servizi in grado di fornire i dati, la logica applicativa e l'interfaccia di cui ha bisogno – per la maggior parte servizi aperti, reperibili nel Web.

Questa è un'attività nuova, specifica dello sviluppo di mashup, che precede la creazione del mashup e assorbe implicitamente l'analisi e la specifica dei requisiti. L'idea iniziale può essere, infatti, vista come un'espressione informale dei requisiti dell'applicazione. I servizi selezionati provano la fattibilità dell'idea e allo stesso tempo esprimono un primo abbozzo di quella che sarà l'organizzazione finale del mashup.

□ *Creazione del mashup*: strumenti "ad-hoc" per lo sviluppo dei mashup devono permettere anche agli utenti meno esperti di combinare alcuni componenti e definire la logica di integrazione necessaria e gli aspetti di presentazione dell'applicazione composita. La logica d'integrazione deve basarsi su formalismi e su modelli intuitivi. I modelli d'integrazione e di esecuzione del mashup possono essere mascherati da notazioni visuali, per semplificare così ancor di più agli utenti meno esperti l'attività di composizione.

Questa attività dà un senso diverso alle tradizionali fasi di *progetto e implementazione*, perché ridimensiona la necessità di eseguire attività tipiche nel mondo del Web, come la

progettazione dell'ipertesto. Il rilascio dell'applicazione consisterà semplicemente nel salvare il mashup e renderlo disponibile su una piattaforma di esecuzione (un'attività che dovrebbe richiedere un solo click).

□ *Uso e manutenzione*: una volta rilasciato sulla piattaforma di esecuzione, il mashup sarà immediatamente pronto per essere utilizzato. La manutenzione dell'applicazione sarà condivisa tra il compositore del mashup (che per esempio risolverà problemi nella logica di composizione) e il gestore della piattaforma di esecuzione (che per esempio risolverà i problemi legati all'ambiente di esecuzione).

Questa attività incorpora le tradizionali fasi di *test e valutazione* e *uso e manutenzione*. Eseguendo facilmente il mashup, il compositore può controllare con altrettanta facilità che l'applicazione funzioni correttamente e che risponda ai suoi bisogni, o addirittura testarla con altri utenti. Eventuali evoluzioni successive richiederanno semplicemente di ripetere il processo di sviluppo, ripartendo dalla prima fase di scoperta e selezione.

L'applicabilità di un tale processo di sviluppo dipende dalla disponibilità di un insieme di strumenti adeguati, tra i quali una piattaforma dedicata per l'esecuzione dei mashup e un insieme di servizi aperti, disponibili nel Web, in grado di fornire nuovo valore attraverso un ricco insieme di funzionalità e di dati. Il processo di sviluppo può essere ulteriormente semplificato nella fase di rilascio e di pubblicazione del mashup se si fa uso di piattaforme in grado di supportare sia la composizione del mashup sia la sua esecuzione (questo in parte è già possibile). In tale contesto di sviluppo, è quindi possibile affermare che il ciclo di vita dei mashup parte dal punto di rilascio nella figura 2 A e procede attraverso evoluzioni incrementali. Infatti, una volta creato, il mashup è immediatamente in esecuzione, senza la necessità di cicli di sviluppo incrementali.

5. STRUMENTI DI SVILUPPO PER I MASHUP

Quali strumenti di sviluppo possono abilitare anche gli utenti meno esperti a sviluppare i propri mashup e a innovare?

L'integrazione di componenti in un mashup può essere realizzata tramite un qualsiasi lin-

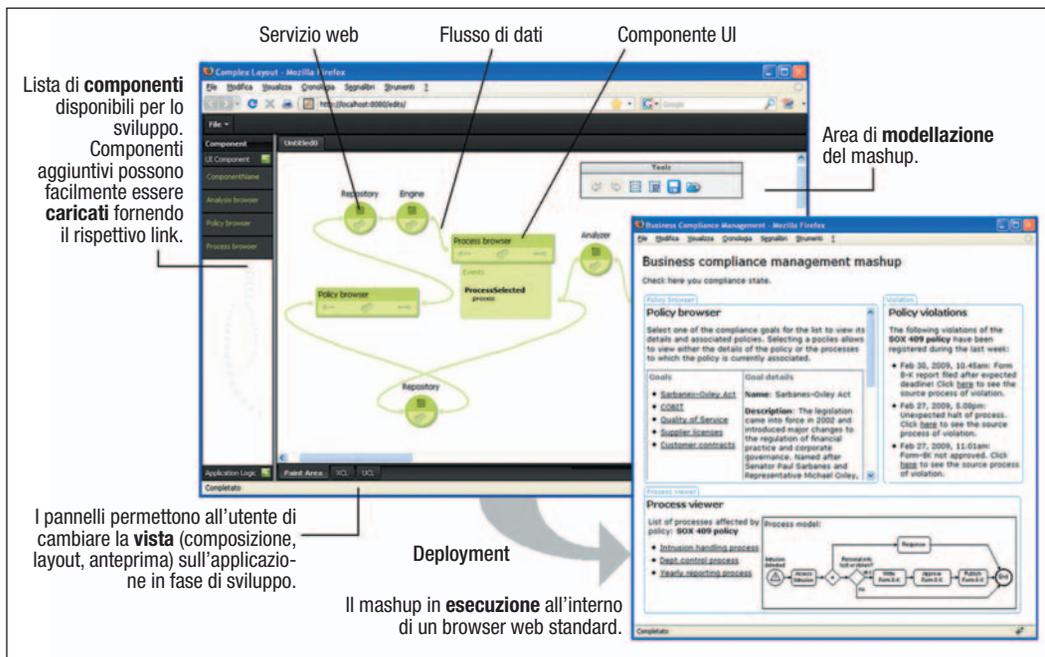


FIGURA 3
 L'editor mashArt per lo sviluppo visuale di "composizioni universali". Le due schermate mostrano la composizione visuale e il mashup finale per un'applicazione di vigilanza sulla conformità alle norme (la cosiddetta "compliance")

guaggio di programmazione, per esempio PHP o JavaScript, tanto per citare i linguaggi più utilizzati [11]. Vista l'eterogeneità dei componenti, dei linguaggi di programmazione, dei protocolli di interazione, della complessità della logica di integrazione, lo sviluppo manuale dei mashup è tuttavia praticabile solo da parte di programmatori esperti ed è sicuramente fuori della portata degli utenti finali del Web. Gli approcci ai BPEL (*Business Process Execution Language*) proposti per la composizione dei servizi non sono in grado di far fronte in modo adeguato all'eterogeneità delle tecnologie che caratterizza i mashup e sono inoltre troppo complessi da utilizzare. Permettere a un'ampia classe di utenti (non solo agli sviluppatori esperti) di comporre mashup, e quindi di innovare, richiede la disponibilità di strumenti di sviluppo intuitivi e facili da usare [2]. Esistono già in letteratura diverse proposte di strumenti per lo sviluppo dei mashup (i cosiddetti "mashup makers"), che in genere mettono a disposizione ambienti di sviluppo visuali per poter facilmente comporre i mashup a partire da una selezione di servizi. Tuttavia, nessuno di questi strumenti offre un ambiente integrato di sviluppo in grado di supportare appieno la composizione di servizi eterogenei. Ognuno di essi offre supporto per specifiche attività di sviluppo. Per esempio, Yahoo Pipes (<http://pipes.yahoo.com>) si concentra sull'in-

tegrazione dei dati (RSS o Atom feed) e offre un linguaggio tramite cui comporre flussi di dati (*pipes*), mentre non fornisce alcun supporto per l'integrazione a livello di presentazione. JackBe Presto (<http://www.jackbe.com>) adotta un approccio per l'integrazione dei dati simile a quello delle pipes di Yahoo, permettendo in più l'aggregazione di componenti visuali (*mashlets*) per la resa dell'output del mashup finale. Nessuno di questi strumenti permette l'integrazione a tutti e tre i livelli che caratterizzano le applicazioni Web, e cioè dati, logica applicativa e presentazione. Lo sviluppo di una piattaforma che copra i tre strati è oggetto della nostra ricerca corrente, il cui risultato è un ambiente di sviluppo e di esecuzione dei mashup, *mashArt* (<http://mashart.org>)², che ambisce a supportare una *integrazione universale*, cioè un approccio di integrazione in grado di gestire in modo trasparente una varietà di tipi di componenti e di tecnologie, dai semplici RSS feed ai più complessi servizi SOAP e RESTful e ai componenti UI. Come mostrato nella figura 3, in *mashArt* la composizione dei mashup è eseguita per mezzo di un editor visuale, sviluppato in AJAX. Gli utenti

² *mashArt* è l'evoluzione di *Mixup* [15], uno strumento precedentemente sviluppato per l'integrazione al solo livello di presentazione.

possono tracciare visualmente la logica di composizione dei loro mashup e definire il layout dell'applicazione finale. Possono quindi salvare e caricare la loro applicazione sulla piattaforma, ed eseguirla istantaneamente col supporto della piattaforma in modalità "hosted"; tutto senza dover scrivere una sola riga di codice. La logica di composizione è basata su eventi (tramite cui è gestita la sincronizzazione delle interfacce utente di ogni componente), e sul flusso dei dati (in base al quale è possibile orchestrare l'esecuzione dei servizi)³.

6. SVILUPPI FUTURI

In questo articolo, abbiamo proposto una visione sui mashup, sugli strumenti per il loro sviluppo, e sul loro ciclo di vita, nel tentativo di contribuire alla definizione di una base comune di concetti e di metodi per lo sviluppo di tali applicazioni. È nostra opinione che una comprensione sistematica di come le pratiche correnti per lo sviluppo di applicazioni Web debbano modificarsi per rispondere alle nuove esigenze dei mashup sia necessaria, soprattutto considerando che il fenomeno dei mashup è destinato a crescere sia nel contesto aziendale [7], sia in quello più ampio degli utenti del Web [15].

Abbiamo mostrato come abilitare ogni classe di utenti a sviluppare le proprie applicazioni richieda la definizione di metodi facili da comprendere e gestire. Al fine di rendere applicabile il modello di sviluppo leggero discusso in questo articolo, la ricerca sui mashup deve necessariamente concentrarsi sui seguenti aspetti:

□ *servizi facili da usare*: è necessario definire modelli e linguaggi di descrizione intuitivi ed espressivi per i dati, la logica applicativa e le interfacce utente dei servizi, in modo da rendere semplice la scoperta e l'integrazione dei servizi. Funzionalità di ricerca e di raccomandazione per la selezione dei servizi sono altrettanto necessarie.

□ *Strumenti per la composizione visuale*: è auspicabile definire notazioni visuali intuitive per la modellazione dei mashup, tali da poter mascherare l'eventuale complessità del linguaggio di composizione con cui è realmente

gestita l'esecuzione dei mashup, accoppiati a meccanismi per la trasformazione automatica delle notazioni visuali in codice eseguibile dell'applicazione.

□ *Piattaforme di esecuzione adeguate*: vista la natura situazionale dei mashup, il rilascio e la pubblicazione dei mashup devono essere facilitati, così da poter eseguire e testare in tempi brevi le applicazioni create, per esempio tramite un interprete online del linguaggio di composizione. Un contributo in tale direzione può derivare dall'adozione di piattaforme web dedicate all'esecuzione "hosted" di mashup.

□ *Progettazione orientata all'interoperabilità*: i servizi, in primo luogo, e anche i mashup devono essere interoperabili, cioè riusabili su più piattaforme. È perciò auspicabile definire standard specifici per i mashup.

□ *Robustezza dei mashup*: sebbene gli sforzi della ricerca corrente sui mashup si concentrino soprattutto sugli aspetti metodologici e sulle tecnologie abilitanti, sarà necessario affrontare problematiche legate all'affidabilità e alla sicurezza dei mashup – soprattutto per i mashup utilizzati in contesti critici, quali quelli di business.

In fine, è importante sottolineare che, se da un lato i processi di sviluppo leggeri sono necessari per alleviare gli sforzi degli sviluppatori di mashup, e in particolare quelli degli utenti meno esperti, lo sviluppo dei servizi da integrare nei mashup rimane un'attività che richiede competenze tecniche specifiche e che deve essere eseguita secondo le pratiche di sviluppo tradizionali. Infatti, se da un lato il successo di un mashup è influenzato dal valore aggiunto dato dalla combinazione di servizi diversi, dall'altro la qualità della composizione finale dipende fortemente dalla qualità di ogni singolo servizio. La definizione di modelli e di tecniche per lo sviluppo di servizi di qualità e per la valutazione della loro qualità è quindi un'altra promettente direzione verso lo sviluppo di mashup di qualità [3], che può ancor più promuovere l'innovazione guidata dagli utenti.

Bibliografia

- [1] Balasubramaniam S., Lewis G.A., Simanta S., Smith D.B.: *Situated Software: Concepts, Motivation, Technology, and the Future*. IEEE Software, Nov-Dec, 2008, p. 50-55.

³ Una dimostrazione di mashArt è disponibile all'indirizzo <http://mashart.org/mashArt.wmv>.

