POLITECNICO DI MILANO

Facoltà di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

Flessibilità e dimensione sociale nei processi collaborativi

Analisi, modelli e architetture di riferimento

Relatore: Prof. Maristella MATERA

Tesi di Laurea di:

Riccardo CHIODAROLI matr. 748970

Anno Accademico 2010 - 2011

Indice

Indice delle figure

Introduzione

Le tecnologie *social* hanno trasformato il Web da una piattaforma per la fruizione passiva di contenuti in un luogo in cui gli utenti possono contribuire attivamente alla generazione di contenuti e idee. La prima generazione di questi strumenti era composta perlopiù da blog, wiki e applicazioni per la condivisione di file, cui è seguita rapidamente una seconda, maggiormente incentrata su relazioni interpersonali: i social network, il cui scopo può essere sintetizzato nel facilitare la comunicazione e lo scambio di messaggi tramite la creazione di una rete personale di contatti.

La diffusione del primo Web, d'altro canto, aveva già profondamente segnato l'ambito professionale, spingendo molte industrie ad adattare le proprie applicazioni B2C, B2B e B2E ad un'architettura Web/SOA e ad impiegare interfacce utente web-based. L'avvento del Social Web ha quindi influenzato nuovamente il mondo industriale, con effetti piuttosto evidenti: gli uffici del personale utilizzano già da tempo piattaforme professionali come LinkedIn per la ricerca di canditati e competenze specifiche; i reparti di marketing ricorrono ai social network per fruire degli immensi bacini d'utenza per sondare il mercato o pubblicizzare un nuovo prodotto; il supporto clienti, poi, utilizza sempre più comunemente strumenti di blogging o social network per agevolare la comunicazione con i clienti ed incrementare trasparenza fidelizzazione.

È quindi piuttosto semplice prevedere che le iniziative prima menzionate diventeranno maggiormente sistematiche, coinvolgendo sempre più attivamente l'utenza esterna nei processi gestionali aziendali, e non solo: molte organizzazioni in generale potrebbero trarre grande beneficio da una sinergia tra i propri processi interni e la partecipazione di clienti e utenti.

Obiettivi della tesi

Scopo ultimo di questa tesi è ricercare la possibilità di ottenere un approccio sinergico per la creazione di **processi sociali flessibili** attraverso l'integrazione di servizi esterni esistenti e il supporto di reti sociali pubbliche o private. L'enfasi sarà posta in particolare su *processi collaborativi*, in cui il requisito di flessibilità è particolarmente accentuato. Ma non solo, poiché anche il contributo degli attori stessi, intesi come persone fisiche, assume un ruolo di sempre maggiore importanza.

La trattazione si svolgerà quindi seguendo due dimensioni ortogonali: una prima legata alla flessibilità dei processi tramite l'integrazione di servizi e una

seconda incentrata invece sull'aspetto sociale e il coinvolgimento attivo degli utenti e della comunità di riferimento nello svolgimento delle attività collaborative. Inizialmente verrà presentato un metamodello di riferimento per la definizione dei processi, che predisponga i concetti ed le operazioni fondamentali senza però legarsi ad un formalismo o linguaggio specifico. Saranno quindi introdotti i concetti di *flessibilità*, per prima, e di *dimensione sociale*, per seconda, con l'intento di estendere la definizione di processi prima fornita considerando nuovi aspetti.

La parte successiva s'incentrerà poi sulla definizione di una possibile architettura software per la gestione e la fruizione dei processi sociali flessibili così definiti, e successivamente, gli strumenti teorici proposti saranno applicati ad uno studio di caso reale, relativo al programma di studi dell'Alta Scuola Politecnica; in particolare, saranno analizzati due scenari, che forniranno il contesto utile alla definizione di due processi sociali e flessibili. Termina la fase di analisi, sarà descritta l'implementazione di una soluzione software in grado di concretizzare il sistema proposto. Saranno precisati dettagli circa le scelte architetturali e tecnologiche effettuate e verranno presentati una serie di *screenshot* dimostrativi circa l'effettivo funzionamento del sistema. La discussione si chiuderà quindi evidenziando i risultati ottenuti e descrivendo alcuni possibili sviluppi futuri.

Struttura della tesi documento

Il documento è così organizzato:

Parte I: analisi dello stato dell'arte

1. *Computer Supported Collaborative Work*: propone un breve digressione circa le tematiche di lavoro collaborativo, il concetto di groupware e una sua classificazione; saranno anche presentati alcune problematiche riconosciute.

2. *Business Process Modeling*: in questo capitolo viene illustrato il concetto di business process, e quindi il tema del Business Process Modeling (BPM): obiettivi, linguaggi di modellazione, flusso di gestione e limitazioni della prospettiva classica; introduzione a Social BPM e principali proposte attuale.

3. *Crowdsourcing:* introduce la nozione di *crowdsourcing*, con un'analisi della filosofia e degli scenari più diffusi, con riferimenti ai principali utilizzi e prodotti commerciali.

4. *Cloud computing:* viene presentato il concetto di cloud computing nelle sue diverse accezioni, con particolare riferimento ai principali vantaggi, le architetture e i paradigmi di riferimento (SaaS, PaaS, IaaS).

5. *Social network:* sono analizzate le funzionalità comuni dei social network, quindi è presente una breve descrizione di un insieme di prodotti commerciali piuttosto diffusi ed utilizzati, osservando in particolare i tratti specifici di ognuno.

Parte II: soluzione concettuale

6. *Meta modello di processo:* in questo capitolo sono poste le basi di riferimento per l'intera discussione; più precisamente, viene proposto un meta modello di processo, non legato ad una particolare realtà o scenario, ma sufficientemente completa da non trascurare alcun concetto descrittivo o funzionale.

7. *Dimensione flessibilità:* estende il meta modello appena definito per includere un formalismo in grado di consentire il *mapping* tra attività e servizi web esterni, con l'intento di offrire grande flessibilità nella definizione e nell'esecuzione dei processi, garantendo inoltre una maggiore facilità integrabilità d'integrazione.

8. *Dimensione sociale:* ortogonalmente al capitolo precedente, viene definita formalmente la dimensione sociale dei processi collaborativi

9. *Studi di caso: reali sociali attuali:* propone uno studio dei principali social network in riferimento alla dimensione sociale appena definita, evidenziando le corrispondenze con le rispettive funzionalità e operazioni

10. *Architettura:* presenta un'architettura di riferimento per una soluzione software in grado di concretizzare il problema concettuale proposto.

Sono descritti i componenti coinvolti, con le specifiche funzionalità e le relative interdipendenze.

Parte III: dominio specifico, Alta Scuola Politecnica

11. *Analisi*: questa sezione propone un possibile caso applicativo per il modello concettuale proposto nella Parte II considerando i processi organizzativi in uso per l'Alta Scuola Politecnica. In particolare, due processi sono descritti, definiti formalmente secondo la BPM Notation e ingegnerizzati rispetto ai servizi esterni e social network disponibili.

12. *Realizzazione:* descrive la realizzazione della soluzione software prodotta per lo scenario precedente, con particolare attenzione alle scelte di ingegneria del software effettuate per soddisfare i requisiti specifici di questo contesto.

13. *Conclusioni:* conclude la trattazione riassumendo i risultati prodotti e proponendo una serie di possibili sviluppi futuri, ivi inclusi alcuni scenari alternativi per cui la soluzione potrebbe dimostrarsi efficace.

Appendice

Bibliografia

Parte I
Stato dell’arte

1. Computer Supported Collaborative Work

Il termine computer supported cooperative work è stato utilizzato per la prima volta da Irene Greif and Paul M. Cashman nel 1984. Nel 1987 Charles Findley introduce il concetto di apprendimento collaborativo (collaborative learning-work). Secondo la definizione di Carstensen e Schmidt il CSCW si occupa di "come l'esecuzione e il coordinamento di attività collaborative possono essere supportate da sistemi informatici"

Da un lato alcuni autori tendono a considerare CSCW e groupware sinonimi, altri ritengono che, mentre i groupware corrispondono esclusivamente a sistemi informatici, il CSCW sia riferisce a strumenti e tecniche di lavoro collaborativo, sia ai loro effetti sociali, psicologici e organizzativi. La definizione di Wilson chiarisce in maniera esauriente tale concetto:

> *"CSCW [is] a generic term, which combines the understanding of the way people work*
>
> *in groups with the enabling technologies of computer networking, and associated*
>
> *hardware, software, services and techniques. "*

Tuttora permane parecchia confusione circa il CSCW; questa nasce dalle differenti interpretazioni dei termini collaborazione e cooperazione che, ancora una volta, molti autori ritengono sinonimi, mentre altri individuano distinzioni tra i due. Collaborazione e cooperazione non differiscono tanto per il fatto che il lavoro sia o meno suddiviso tra vari attori, quanto piuttosto per il modo in cui il lavoro viene suddiviso: nella cooperazione, infatti, il lavoro è diviso in task indipendenti, mentre nella collaborazione è il processo cognitivo ad essere suddiviso. Nella cooperazione il coordinamento è quindi richiesto solo quando i vari risultati parziali sono integrati, mentre la collaborazione è un'attività sincrona e coordinata che è il risultato di un tentativo di costruire e mantenere una visione comune di un certo problema.

Il concetto di cooperazione è inoltre sovente associato ai concetti di coordinamento e comunicazione. La suddivisione del lavoro in sotto-attività (task) porta naturalmente all'esigenza di coordinamento, in questo caso il coordinamento può essere visto come "la gestione delle dipendenze tra attività e il supporto delle interdipendenze tra gli attori". La comunicazione è invece il

sistema attraverso il quale l'informazione può essere scambiata tra individui per mezzo di simboli e comportamenti condivisi. "La comunicazione è il collante di un'organizzazione: maggiore è la necessità di coordinamento e cooperazione, tanto maggiore sarà la necessità di comunicare".

Poiché la differenza tra collaborazione e cooperazione è molto labile, e tende a rimanere un'astrazione teorica piena di sfumature che mal si adatta all'ambito applicativo; nel seguito si considereranno collaborazione e cooperazione come sinonimi ai quali verrà dato il significato di cooperazione enunciato sopra.

1.1 Groupware e software per il lavoro collaborativo

Gruppi che sono dislocati in spazi diversi, sia all'interno di uno stesso edificio sia in differenti paesi o in diverse organizzazioni, necessitano di un supporto alla collaborazione che va oltre la condivisione della macchina del caffè o della scrivania. Oggi questa situazione di dispersione è in aumento per l'accresciuta mobilità e per la distribuzione sovranazionale di molte organizzazioni, quindi le applicazioni per il lavoro cooperativo diventano un imperativo in molti ambiti, dal business all'istruzione fino alla ricerca.

Nella categoria dei groupware si collocano svariate applicazioni molte delle quali tra loro eterogenee. I groupware possono essere classificati in tre categorie in base al livello di collaborazione, come illustrato in Figura 1:

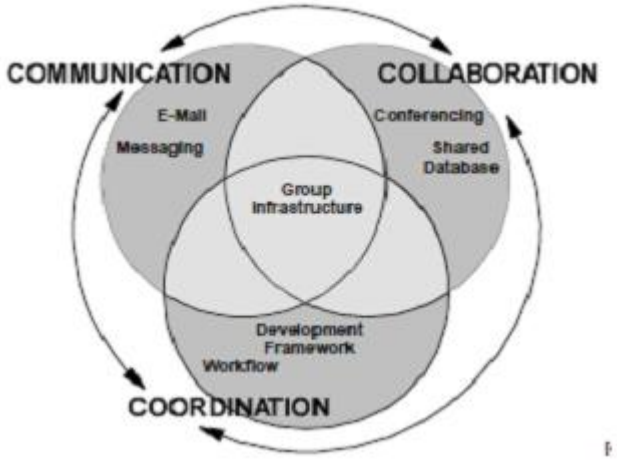


Figura 1 Classificazione dei groupware

- Strumenti di *comunicazione*, come e-mail, instant messaging, fax e wiki che permettono lo scambio di messaggi, documenti e file

- Strumenti di *collaborazione* (conferencing), come forum, chat, video conferenze, application sharing che facilitano lo scambio di informazioni in maniera più interattiva

- Strumenti di *coordinamento*, come sistemi di project management e di gestione dei workflow (Workflow Management Systems, WfMSs) che gestiscono e semplificano le attività che coinvolgono più utenti.

1.2 Problematiche

Mentre alcuni lavori presenti in letteratura hanno dimostrato come l'efficienza del lavoro di gruppo migliori facendo uso di processi collaborativi strutturati, molti prodotti software esistenti sono decisamente "task-oriented" e non "process-oriented" [26]: più precisamente, si focalizzano sull'esecuzione di singolo *task*, offrendo in compenso uno scarso supporto per i processi collaborativi, tipicamente per mezzo di funzioni di cooperazione già codificate e scarsamente configurabili, limitando così le possibilità degli attori di essere pienamente coinvolti nel processo.

Diversamente, i WfMSs sono normalmente troppo rigidi per garantire la vera flessibilità dei processi collaborativi: la definizione del processo deve essere completata in fase di progettazione, poiché una volta avviato approntare eventuali correzioni e modifiche può rivelarsi destramente complesso.

Un approccio intermedio corrisponde invece agli *evolving workflows* [25][27], che consentono di create processi *parzialmente definiti*, ovvero processo in cui solo una parte del flusso di esecuzione può essere stabilita in fase progettuale, e per il quale la specifica completa delle attività presenti, unitamente alla loro sequenza, è nota soltanto a *run time*. In [24] viene proposta una caratterizzazione dell'evoluzione di workflow tramite tre dimensioni: *dinamismo*, inteso come la necessità di modificare i modelli soggiacenti i processi nel momento in cui il business process relativo sia modificato (ciò crea difficoltà specialmente con processi ancora attivi, la cui consistenza può venire compromessa); *adattabilità*, definita come la capacità di un processo di tollerate eccezioni (per quanto molte di queste possano essere previste in progettazione, non è possibile prevenire qualsiasi situazione irregolare); *flessibilità*[1], come menzionato in precedenza il processo è definito completamente soltanto in fase di esecuzione.

[1] L'accezione di flessibilità qui intesa non coincide con la proposta del capitolo 8

2. Business Process Modeling

Con il termine Business Process Modelling (BPM) s'intende l'attività di rappresentare e definire processi aziendali, comparandone le versioni attuali (*as is*) e proposte (*to be*) nell'ottica di favorire un'analisi del contesto lavorativo e dare luogo, in ultimo, a miglioramenti operativi e d'efficienza. Sebbene l'attività in sé non appartenga propriamente all'ambito delle tecnologie informatiche, queste ne costituiscono di fatto un'importante catalizzatore, specialmente vista la grande pervasività dell'IT nel mondo industriale moderno. Prova né anche il gran numero di soluzioni software dedicate all'argomento, che annovera anche noti brand del mondo IT come IBM o Oracle.

2.1 Business Process

Un *business process* è un insieme di attività strutturate e correlate, con lo scopo di erogare uno specifico servizio o produrre un determinato bene verso uno o più clienti. Solitamente, si distinguono tre tipologie di business process:

- **Managament process:** corrispondono a processi che determinano la gestione dei sistemi produttivi; ne fanno parte processi come la *corporate governance* e lo *strategic management*.

- **Operational process:** costituiscono il *core business* dell'azienda e riguardano il flusso primario di produzione. Ne sono esempi l'acquisto di materiali presso i fornitori, la produzione, il marketing e la vendità

- **Supporting process:** come suggerisce il nome, agiscono in supporto di altri processi; non contribuiscono direttamente alla produzione di valore e soddisfano invece requisiti interni: la contabilità, il supporto tecnico e il reclutamento sono tutti esempi di processi di supporto.

La definizione di business process avviene normalmente per mezzo di linguaggi o notazioni standard o ampiamente utilizzate nel settore[2]. Alcune, come il ben noto UML o la BPM Notation (vedi Appendice) hanno un approccio maggiormente teorico e astratto, senza alcun legame verso tecnologie o

[2] http://www.bpmodeling.com/faq/

soluzioni concrete; altre, come ad esempio BPEL (Business Process Executable Language) o WS-CDL (Web Service Choreography Description Language) sono invece standard pensati per un approccio più sinergico tra la definizione, l'*orchestration* e l'integrazione con soluzioni software esistenti.

2.2 Ciclo di vita

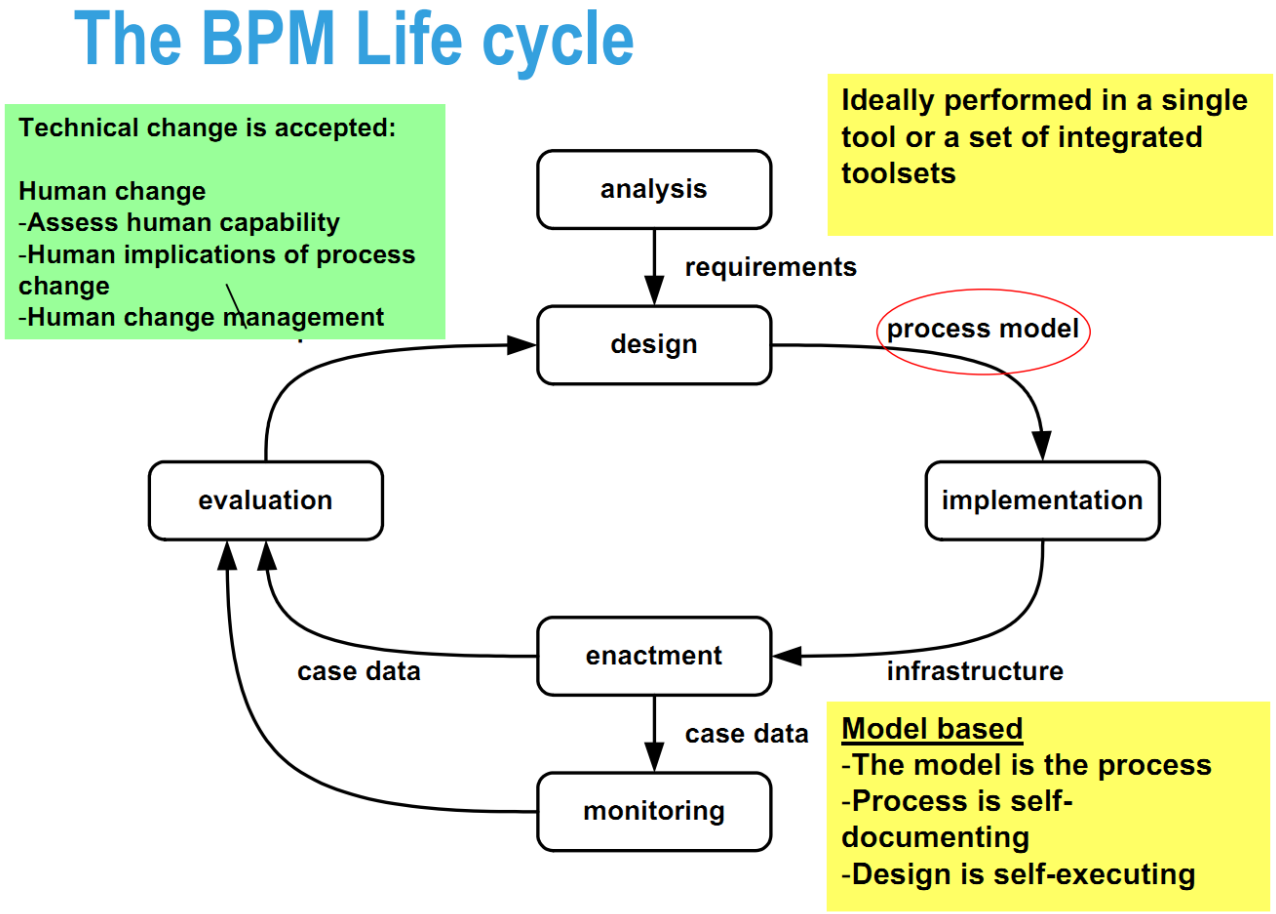


Figura 2 Ciclo di vita nella soluzioni BPM (Balbir Barn 2007)[3]

In Figura 2 è mostrato un tipico ciclo di vita nell'ambito di soluzioni BPM: a seguito di una prima fase di analisi, che permette di definire i requisiti di processo essenziali. Le metodologie adottate in questo caso non differiscono particolarmente da altre a in uso, per esempio, nella progettazione di software: di norma viene prodotta una prima analisi, volta a definire correttamente il dominio operativo e il contesto d'esecuzione del processo, con particolare attenzione all'identificazione degli stakeholder. Sarà infatti necessario coinvolgere questi ultimi tramite il noto meccanismo delle interviste, di

[3] http://www.jisc.ac.uk/media/documents/programmes/eframework/process-modelling_balbir_barn.pdf

fondamentale importanza per precisarne il coinvolgimento pratico, il ruolo, le operazioni svolte con relativo "output" e gli eventuali altri attori coinvolti.

Segue poi la progettazione vera e propria, dove il processo viene strutturato e definito tramite una sintassi o linguaggio tra quelli citati in precedenza. Successivamente il processo viene implementato e posto in uno scenario d'esecuzione. Tuttavia il ciclo non si esaurisce a questo punto, dando vita ad una serie di fasi di valutazione dell'efficacia della soluzione prodotta che contribuiscono al continuo miglioramento della gestione di processo.

1.1.1 Limitazioni e punti di criticità

Il ricorso ad un approccio di BPM chiuso nell'ambito di processi destinati a clienti o cittadini, o anche in ambito aziendale per scenari non ripetitivi e fortemente basati sulla comunicazione tra attori, può generare diversi problemi e inconvenienti: ad esempio, forzare gli utenti finali a seguire un modello e flusso operativo rigido, invece che coinvolgerli attivamente nella definizione del processo e permettere loro di fornire la propria opinione o *feedback* sull'usabilità e fruibilità del servizio, genera tipicamente un forte sentimento d'insoddisfazione potenzialmente lesivo.

Un altro aspetto da non sottovalutare, poi, è la tendenza diffusa in molte organizzazioni di proporre un'eccessiva strutturazione e rigidità, con il risultato che spesso gli attori coinvolti nei processi aziendali "evadano" dagli schemi e svolgano il proprio compito secondo modo e tempi non previsti; fatto, questo, che inficia anche tematiche secondarie come la valutazione dell'efficienza produttiva e il possibile miglioramento dell'operatività interna. Una simile circostanza può presentarsi anche in contesti ben più complessi e delicati, quali sono i casi di riorganizzazioni dovute a fusioni o acquisizioni, in cui l'integrazione di realtà prima distinte in un solo blocco organizzativo potrebbero essere fortemente semplificate dalla capacità di coinvolgere gli attori dei processi.

2.3 Social BPM

Lo straordinario successo dei social network negli ultimi anni ha mutato, permanentemente, il modo in cui gli utenti comunicano online, al punto che sempre più impiegati, clienti e persone comuni si aspettano di poter usufruire di qualsiasi servizio nelle stesse modalità di condivisione delle informazioni e con il medesimo livello di coinvolgimento. Si pensi ad esempio all'enorme numero di commenti o voti generati ogni giorno, su notizie, idee, opinioni, servizi o anche

prodotti e aziende. Queste ultime, in particolare, hanno da poco iniziato ad impiegare soluzioni social nel proprio business tradizionale, secondo però filosofie diverse: da un lato molte imprese ricorrono a prodotti popolari, per migliorare la propria visibilità (Facebook e Twitter sono i canali tipici) o anche per ricercare candidati e competenze specifiche per mezzo di servizi specializzati come LinkedIn. Al contrario, alcune realtà hanno scelto di rivedere i propri processi interni in ottica sociali, interpretandone personalmente l'approccio e i vantaggi: ne è un esempio Nokia con l'iniziativa Design By Community, che recentemente ha chiesto direttamente ai propri clienti e utenti di realizzare la nuova suoneria per i propri telefoni e dispositivi mobili. In ultimo, è bene sottolineare come la grande maggioranza, se non la totalità, delle soluzioni sociali social esistenti disponga di API (Application Programming Interface) per favorire l'integrazione dei propri servizi all'interno di software di terze parti.

A fronte dei problemi prima evidenziati in ambito BPM e di quanto ora affermato, appare piuttosto logico come sia nata una nuova accezione di BPM concepita sinergicamente con soluzioni social, Social BPM. SI tratta tuttavia di un ambito piuttosto recente e ancora poco studiato; il termine stesso è attribuito a Clay Richardson, che lo impiegò per la prima volta nel Novembre 2009[4], mentre un primo studio specifico risale al maggio 2010, a cura di Gartner Research[5]. In seguito, il tema ha acquisito sempre maggiore interesse, portando anche alla pubblicazione di svariati articoli in merito. In [Fraternali, Brambilla], ad esempio, è stato proposto uno studio che prevede una possibile estensione della BPM Notation in modo da supportare l'integrazione di social network a livello di process design.

Stranamente, però, ad oggi sono ancora pochi i riscontri in ambito industriale, in particolare quelle soluzioni software allo stato dell'arte nell'allo stato dell'arte nell'ambito BPM tradizionale, che appaiono poco propense ad adottare nuovi approcci e metodologie. A tal proposito il prof.Marco Brambilla ha affermato:

> *despite some criticism on the term "Social BPM", I would say there is wide consensus on the need of integrating rigid BPM approaches with*

[4] https://www-950.ibm.com/events/wwe/grp/grp004.nsf/vLookupPDFs/Forge%20Your%20Smart%20Work%20Game%20Plan%20Clay%20Richardson/$file/Forge%20Your%20Smart%20Work%20Game%20Plan%20Clay%20Richardson.pdf

[5] http://blogs.gartner.com/jim_sinur/2010/05/05/social-bpm-design-by-doing/

others that consider user interactions as crucial value for the enterprise. To my surprise, several experts agree that there is a substantial request for these technologies by customers, especially in "non-traditional" scenarios. On the other side, the state of the art of the tools and systems is still perceived as weak or only partially addressing the actual needs.

Uno studio svolto in [BPM4ForPeople] ha evidenziato come la maggioranza dei prodotti BPM di prima fascia siano eccessivamente complessi, costosi, privi di funzionalità sociali di base e dotati di un'architettura monolitiche che rende estremamente complessa l'integrazione di una qualsiasi soluzione social esistente. Oltretutto, soltanto un numero limitato di produttori (tra cui IBM, Oracle, Intalio e Software AG) ha annunciato l'intenzione di fornire tale integrazione nelle future edizioni dei propri prodotti.

3. Crowdsourcing

Il termine *crowdsourcing* è stato coniato per la prima volta da Jeff Howe in un articolo della nota rivista Wired nel giugno 2006, dal titolo "The Rise of Crowdsourcing". Howe spiegò che grazie ai recenti progressi tecnologici, unitamente alla diffusione di dispositivi elettronici più economici, il divario tra utenti professionali e semplici amatori si fosse ridotto, fatto, questo, che avrebbe permesso alle aziende di avvantaggiarsi del potenziale "computazionale" offerto dal grande pubblico. A differenza del normale outsourcing, infatti, il crowdsourcing propone di esternalizzare una o più attività ad una comunità o a un gruppo di persone arbitrariamente grande e geograficamente distribuite, in modello di collaborazione aperta. In alternativa:

> *Crowdsourcing is channelling the experts' desire to solve a problem and then freely sharing the answer with everyone*
>
> Henk van Ess, settembre 2010

3.1 Analisi

Le potenzialità del crowdsourcing derivano essenzialmente da due fattori principali: in primo luogo, la mente umana dispone di capacità di risoluzioni di problemi che spesso agisco come complemento della tradizionale elaborazione algoritmica, garantendo minori difficoltà e migliori risultati in determinate e particolari situazioni; ad esempio, il riconoscimento di soggetti ed elementi costitutivi nelle immagini è da sempre un problema aperto: da un lato sono noti problemi di complessità algoritmica e requisiti di calcolo, che tradizionalmente ne circoscrivono l'impiego a scenari server e HPC (high performance computation). Dall'altro esistono però anche difficoltà nei risultati stessi, che spesso risultano imprecisi o addirittura errati. Tutto ciò senza considerare anche ovvi problemi di tipo semantico: una ricerca di fotografie per "jaguar" deve riferirsi al felino o alla casa automobilistica?

Secondariamente, il ricorso ad un vasto insieme di individui (in teoria addirittura illimitato) permette di ricorrere con migliori risultati ad approcci di tipo statistico; si pensi ad esempio ad iniziative di beta testing pubblico: per quanto sia comunemente riconosciuto che la verifica di software tramite il suo utilizzo effettivo rappresenti un metodo più efficace per l'individuazione di bachi, un programma limitato ad un numero ristretto di utenti ha scarsa

possibilità di portare alla segnalazione di problemi rari o di difficile individuazione. Viceversa, con un'ampia comunità di beta testers, la probabilità che un certo bug si verifichi almeno una volta presso almeno un'utente aumenta.

Nella fattispecie, il termine sottintende molti scenari differenti, tra cui la *human computation* , il *participatory design*, e la *citizen science*. È già stato fatto cenno del primo, mentre nel caso di participatory design, l'idea essenziale è il coinvolgimento diretto degli stakeholder nei processi di analisi e progettazione. Il principale vantaggio offerto consiste in rapporto più stretto con le persone interessate dai processi in questione, che possono così contribuire già in fase di studio, e ben prima della realizzazione. In realtà non si tratta di un'idea completamente nuova, ma nell'ambito delle scienze informatiche è stata recentemente riprese in virtù dei grandi cambiamenti, non solo tecnologici, avvenuti, tra cui certamente il Web 2.0. Parlando di citizen science, invece, s'intende il coinvolgimento di cittadini o persone comuni nel più particolare ambito della ricerca, spesso con lo scopo di far fronte ad attività di grandi moli di dati. L'aspetto particolare è il fatto che non vengono richieste competenze specifiche o sistematiche, laddove infatti i compiti assegnati a ciascun individuo sono sufficientemente semplici da essere svolti rapidamente e permettono di estendere la comunità di individui coinvolti nel progetto. Volendo paragonare queste soluzioni ai più noti e diffusi progetti di distributed computing, che si fondano a loro volta, almeno in parte, sul noto proverbio *divide et impera*, si potrebbe citare:

> *Galaxy Zoo volunteers do real work. They're not just passively running something on their computer and hoping that they'll be the first person to find aliens. They have a stake in science that comes out of it, which means that they are now interested in what we do with it, and what we find.*

Kevin Schawinski, astrofisico presso l'università di Yale e fondatore del progetto Galazy Zoo[6]

3.2 Sfide e limitazioni

La proposta del crowdsourcing non è esente da ostacoli e limitazioni oggettive: innanzitutto il coinvolgimento attivo di una vasta comunità di individui non può avvenire casualmente e necessita quindi di essere promosso e

[6] http://www.wikinomics.com/blog/index.php/2009/02/09/crowdsourcing-versus-citizen-science/

gestito accuratamente. Al contempo, non del tutto corretto ipotizzare che il coinvolgimento in progetti o attività possa avvenire spontaneamente, senza ad esempio prevedere alcune forme di remunerazione. In tal senso, è piuttosto interesse una recente proposta nata dai risultati condotti da uno studio Nielsen[7] circa l'ammontare di ore speso, nei soli Stati Uniti, per i giochi online: 407 milioni di ore. La proposta consiste nello sviluppo di una nuova generazione di giochi che funga in realtà da interfaccia grafica per l'esecuzione di attività in progetti di crowdsourcing, come il già citato esempio di riconoscimento di persone, animali o cose all'interno di fotografie. Il gioco dovrebbe prevedere un sistema di punteggi che incentivi all'uso, oltre che allo sviluppo di una comunità di appassionati tramite un meccanismo simile alle *leaderboards* (classifiche dei punteggi migliori). Un progetto già attivo e largamente usato, sebbene non inteso come gioco, è invece il progetto reCaptcha di Google, che unisce la necessità di fornire un servizio per la generazione di captcha alle problematiche di validazione dei risultati dei sistemi OCR, almeno per i casi d'incertezza. In questi casi, il sistema costruisce un captcha sulla base del testo non riconosciuto e incrementa il punteggio base di: 0.5 punti per il valore di ogni software OCR utilizzo, e un punto per ogni riconoscimento umano. Non appena un'identificazione raggiunge un punteggio di 2.5 punti viene considerata corretta.

La soluzione prima esposta suggerisce anche un ulteriore aspetto critico, ossia la *trustability* dei risultati forniti dalle persone coinvolte: di base, non esiste infatti alcuna garanzie che l'operatore svolga onestamente il compito assegnato, e in molti casi una valutazione statistica dei risultati forniti potrebbe rivelarsi insufficiente. Per tale motivo, diverse ricerche si sono focalizzate su questo particolare tema nel tentativo di garantire maggiore "fiducia" nell'output degli utenti.

In ultimo, anche l'interpretazione dei risultati forniti dagli utenti costituisce un potenziale problema: a differenza degli elaboratori e del software tradizionale, infatti, una persona potrebbe fornire una risposta ambigua o non interpretabile da un elaboratore (potrebbe essere il caso di risposte audiovisive). In tal senso, si avverte la necessità di garantire una migliore corrispondenza tra i due mondi, sfruttando sì le capacità di elaborazione umane e garantendo al contempo una più semplice raccolta dei risultati prodotti.

[7] http://blog.nielsen.com/nielsenwire/online_mobile/what-americans-do-online-social-media-and-games-dominate-activity/

3.3 Esempi di prodotti e progetti

In tal senso, prodotti commerciali quali Amazon Mechanical Turk[8] forniscono una soluzione in proposito, proponendo da un lato una serie di *Human Intelligence Tasks* (HITs) predefiniti, quali Data Cleansing, Categorization, Content Moderation e Business Feedback, che dispongono già di un'infrastruttura adatta alla raccolta strutturata dei risulti; dall'altro permettono comunque ai *requesters* di creare le proprie HITs seguendo le guidelines pubblicate. L'interesse per questo prodotto nasce anche a causa del numerod i 500.000 "lavoratori" attivi, in oltre 190 paesi nel mondo, per i quali è previsto un pagamento al completamento dell'HIT scelta. Altro punto di interesse, è poi un meccanismo di Qualification che i requesters possono adottare per richiedere una preparazione rapida prima di affrontare un task, al fine di migliorarne l'esecuzione. Infine, è importante segnalare come ogni worker registrato disponga di una propria reputazione, migliorabile svolgendo correttamente le HITs accettate, ma anche peggiorabile nel caso in cui il requester rifiuti l'operato svolto. Ciò contribuisce a rendere il sistema più equilibrato e a scoraggiare utilizzi malevoli del prodotto.

Una menzione spetta anche ad un progetto di crowdsourcing avviato per mezzo di Mechanical Turk, volto alla ricerca di persone scomparse: per esempio, quando Jim Gray scomparve nel 2007, il suo personale amico di Amazon, Werner Vogels, prese accordi con DigitalGlobe per la fornitura di immagini satellitari aggiornate delle isole Farallon, vicino San Francisco, luogo della probabile scomparsa. Le mappe così ottenute furono caricate su Mechanical Turk e distribuite ai workers per un'analisi di "oggetti estranei". Un caso analogo avvenne poi nel caso della scomparsa dell'americano Steve Fossett[9].

[8] https://www.mturk.com/mturk/welcome
[9] http://www.wired.com/software/webservices/news/2007/09/distributed_search

4. Cloud computing

Dare una definizione di *cloud computing* è sempre piuttosto complesso, anche in virtù dei diversi aspetti potenzialmente attinenti il tema. Tuttavia, la definizione in [BerkleyCloud] è sufficientemente completa e concisa:

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services.

Nella fattispecie, questa definizione sottolinea come il termine cloud assuma significati diversi a seconda dell'attore coinvolto: per gli utenti finali rappresenta un nuova modalità di fruizione delle applicazioni software, non più installate localmente e perciò legate ad una particolare postazione, bensì accessibili remotamente attraverso una connessione di rete. Per le aziende invece rappresenta un'opportunità di gestione flessibile delle proprie risorse IT, in termini di allocazione dinamica, di affidabilità e qualità del servizio e riduzione dei costi.

4.1 Vantaggi principali

L'impiego del cloud consente di ottenere una serie di benefici non trascurabili, che possono essere sintetizzati nei seguenti punti:

- **Flessibilità:** ossia la possibilità di scalare le proprie applicazioni in base a requisiti variabili; più precisamente, è possibile allocare dinamicamente la quantità e la tipologia di risorse assegnate ad una determinata applicazione considerando il carico istantaneo o previsto, incrementando la capacità computazionale in condizioni di alto traffico e diminuendola subito dopo.

- **Programmabilità:** le soluzioni cloud sono progettate per offrire molte funzionalità e servizi di supporto allo sviluppo di prodotti personalizzati, facilitandone al massimo l'integrazione. Viene però garantita anche un'astrazione dal sistema hardware e software sottostante con l'intendo di garantire prestazioni e scalabilità, in modo che non sia necessario riprogettare la parte software.

- **Autogestione:** la totalità delle soluzioni cloud permette di gestire autonomamente la configurazione, il monitoraggio e la manutenzione delle risorse allocate per mezzo di pannelli web, senza bisogno di intermediari o contatti con personale addetto, in modo da semplificare la maggior parte dei bisogni più comuni e velocizzare le tempistiche per la fruizione del servizio.

- **Costi:** ulteriore aspetto di rilievo consiste nelle politiche di tariffazione offerte dai provider, che consentono un'accurata misurazione dei consumi di risorse, in modo che gli utenti paghino esclusivamente in base alla capacità computazionale effettivamente utilizzata. In quest'ambito rientrano poi anche le problematiche di qualità del servizio, per mezzo di SLA (Service Level Agreement) che permettono di richiedere una maggiore affidabilità e garanzie di qualità, a fronte di una tariffazione maggiore.

4.2 Paradigmi

Nell'ambito cloud rientrano in realtà diverse architetture software, che specificano un modello funzionale piuttosto preciso, definendo i componenti coinvolti e l'interazione tra questi. Nel seguito quindi saranno presentati i paradigmi architetturali più comuni e consolidati.

4.2.1 Software-as-a-Service (SaaS)

Il primo paradigma analizzato è noto come **Software-as-a-Service (SaaS)**, in alcuni casi definite anche *software-on-demand*. La caratteristica fondamentale di questa soluzione è di non installare applicazioni e software su dispositivi e postazioni client, erogando invece le medesime funzionalità e operazioni tramite servizi remoti. In tal senso, i client devono solamente interfacciarsi con il servizio (non qui applicazione) e scambiare una serie di comandi per mezzo di messaggi attraverso una rete. In questo modo i requisiti computazionali divengono competenza della parte server, abilitando di conseguenza i *thin client*, quali dispositivi mobili o netbook, di beneficiare di ulteriori applicazioni software prima precluse a causa delle scarse prestazioni. Inoltre, gli utenti sono tenuti a pagare solamente per l'utilizzo effettivo del servizio, invece che acquistare una licenza completa.

Le funzionalità esposte da un servizio possono variare considerevolmente: alcuni sono progettati per scenari di integrazione con altri prodotti (ne sono un esempio i servizi di previsione meteorologica o le quotazioni azionarie), mentre altri si focalizzano maggiormente sugli utenti (si pensi a GMail o DropBox). Ovviamente esistono sia servizi pubblici che privati, intesi per un ambito strettamente aziendale, così come soluzione gratuite e a pagamento. In ogni caso, è pratica comune prevedere un meccanismo registrazione e autenticazione per l'utilizzo di un servizio, anche pubblico, per monitorarne l'utilizzo da parte dei client: alcuni provider, ad esempio, impongono delle quote di utilizzo per l'uso gratuito del sistema e negano ulteriori accessi una volta superate le soglie stabilite.

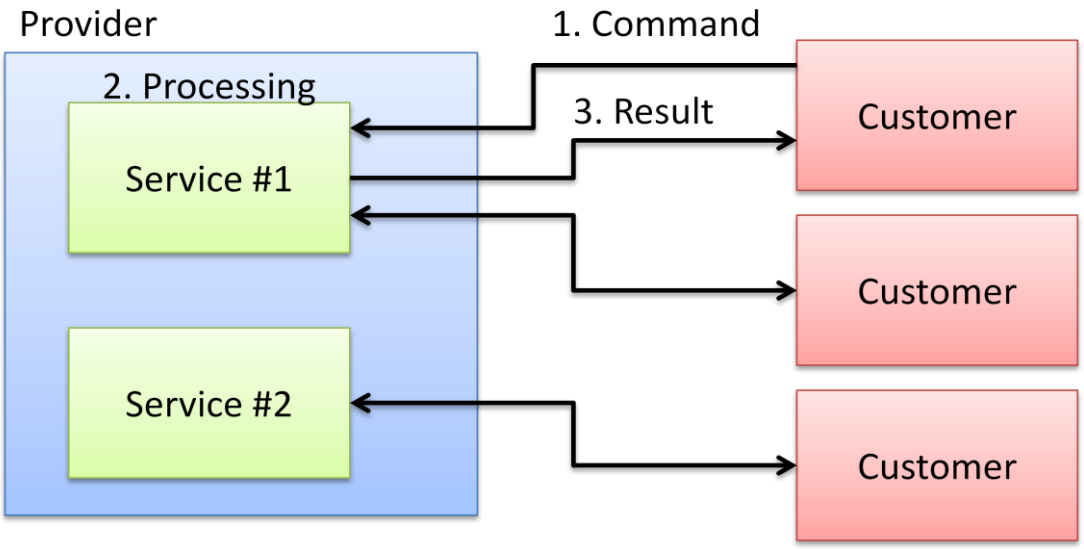


Figura 3 Schema funzionale del paradigma SaaS

La Figura 3 mostra lo schema funzione del paradigma SaaS: da un lato è presente il provider, con i relativi servizi disponibili, dall'altro sono presenti i clienti. Questi ultimi, una volta registrati ed autenticati, comunicano con il servizio via rete, inviano messaggi che identificano le operazioni o i comandi da eseguire. Il servizio remoto svolge quindi l'azione richieste, sfruttando le risorse a disposizione allocate per il cliente, e restituisce il risultato. È significativo notare come questo tipo di architettura fornisca un conoscenza piuttosto limitata del sistema software su cui si basa il servizio remoto. In altre parole, i clienti possono accedervi tramite un'interfaccia nota e stabile, senza preoccuparsi delle problematiche di installazione e manutenzione, di competenza del provider. Per questa ragione, l'integrazione di servizi, forniti anche da provider differenti, è grandemente semplificata e decisamente più gestibile anche in realtà di dimensione medio –piccola.

4.2.2 Infrastructure-as-a-Service (IaaS)

A differenza del caso precedente, che consente agli utenti di accedere al solo servizio, il paradigma **Infrastructure-as-a-Service (IaaS)** riduce il livello di astrazione in modo permettere piena visibilità dei livelli software sottostanti, a partire dal sistema operativo. I clienti possono quindi gestire completamente la piattaforma software per l'esecuzione della proprie applicazioni secondo i requisiti specifici, eliminando però la necessità di contattare un supporto tecnico per le principali attività amministrative.

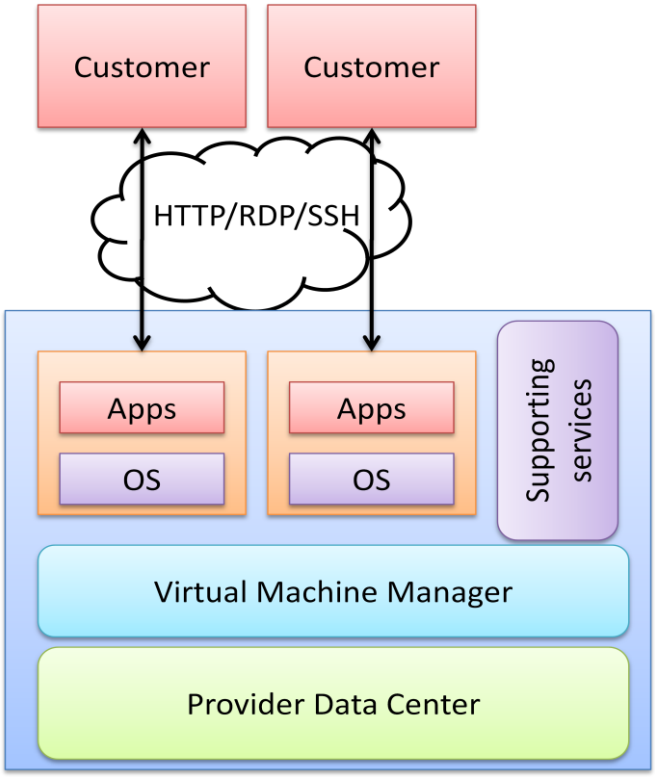


Figura 4 Schema funzionale del paradigma IaaS

La Figura 4 mostra una panoramica funzionale dello IaaS: i provider gestiscono un'infrastruttura di virtualizzazione molto flessibile e di grandi dimensioni nel propri data center e offrono ai propri client la possibilità di configurare ed eseguire macchine virtuale all'interno di essa. Di norma sono disponibili una serie di VM (virtual machines) preconfigurate, con diversi SO e specialmente differenti capacità in termini di computazione. Nella fattispecie, a differenza dei normali server, le cui prestazioni sono vincolati dai componenti hardware installati, le macchine virtuali sono eseguite su *hardware e server virtuali*. Impiegando server di grande capacità, i provider sono in grado di gestire un gran numero di VM in contemporanea e di riservare una parte delle risorse fisiche disponibili per ciascuna, secondo le richieste dei clienti. Ma dal momento che esiste un livello di astrazione sullo hardware reale, è anche

possibile avviare altre VM su richieste, anche con il sistema attivo e in uso. Questa soluzione permette di risolvere innumerevoli problemi, a cominciare dall'incremento dinamico di capacità computazionale in condizioni di alto traffico non atteso (picchi imprevisti). Logicamente, è poi possibile riportare il consumo di risorse al livello precedente non appena il picco di traffico si è esaurito, in modo da non sprecare alcunché.

La possibilità di dimensionare dinamicamente le macchine virtuali ha un impatto significativo anche sui contratti con i clienti, non più obbligati ad acquistare server fisici in modo da tollerare gli scenari di caso pessimo. Grazie allo IaaS, infatti, è possibile allocare una quota base di risorse per supportare una condizione di traffico regolare e richiedere eventualmente un'operazione di *upscaling* nel caso in cui si registri o si attenda un picco di traffico.

4.2.3 Platform-as-a-Services (PaaS)

Esiste infine un approccio ibrido tra SaaS e IaaS, chiamato *Platoform-as-a-Service*. Il punto centrale di questo paradigma consiste nel fornire agli utenti un'astrazione intermedia sull'infrastruttura interna del provider in moda permettere la progettazione e la distribuzione di applicazioni personalizzate senza richiedere però l'installazione e configurazione di software di base come web server o database. Tipicamente, l'accesso a detta astrazione è controllato da una serie di API specifiche da provider a provider, cui corrisponde normalmente anche un ambiente di sviluppo dedicato.

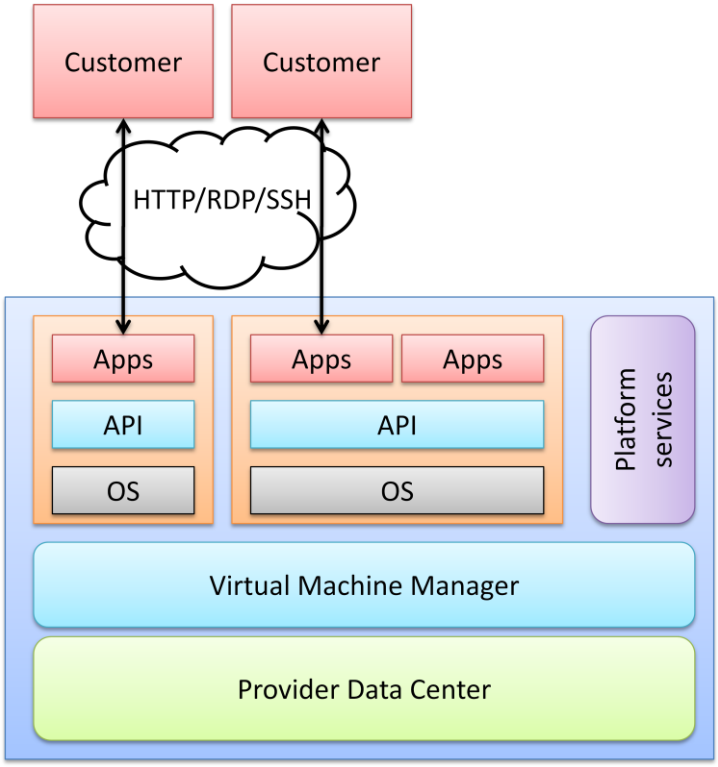


Figura 5 Schema funzionale del paradigma PaaS

La Figura 5 appare molto simile alla precedente, ma esistono delle differenze sostanziali da sottolineare: per prima cosa, il livello di SO non è più visibile da parte delle applicazioni; al contrario, un nuovo livello software, indicato generalmente come API, è previsto per esporre una serie di librerie sulle quali i software sviluppati debbono basarsi per interagire con l'infrastruttura del provider. Ad esempio, alcuni prodotto consentono di gestire il parallelismo solo in questo modo. Meccanismo di archiviazione e persistenza dei dati sono a loro resi disponili attraverso altre API.

Secondariamente, la quasi totalità dei provider non garantisce che un data applicazione sia posta in esecuzione su un server dedicato, o anche solo una macchina virtuale. È quindi piuttosto frequente che più applicazioni condividano direttamente anche risorse virtuali. In ultimo, laddove in precedenza si indicava la presenza di un blocco *supporting services*, nell'ambito del PaaS si preferisce parlare di *platform services* per sottolineare come questi ultimi afferiscano ad uno strato software superiore e sia integrabili nelle applicazioni degli utenti.

5. Social network

Si vuole ora proporre una descrizione dei social network, a partire dagli aspetti fondanti e dalle funzionalità più comuni, passando poi ad una panoramica dei servizi oggi più diffusi e popolari, oppure particolarmente significativi a livello concettuale per obietti e funzionalità proposte.

5.1 Funzionalità tipiche di un social network

Per quanto comunemente note, è comunque opportuno precisare un elenco di funzionalità tipicamente rese disponibili da servizi di social networking:

- **Profili utente**
  Uno dei tratti distintivi, prevede l'inserimento di informazioni "personali" di diverso carattere, a seconda del contesto, in modo che altri utenti possano accedervi, limitatamente alle impostazioni di privacy decise dal soggetto. Classici esempi di informazioni riguardano l'elenco dei contatti, l'affiliazione o il posto di lavoro e un'immagine personale. **NOTA:** il termine non va confuso con un analogo concetto derivato dai sistemi di *user profiling*. Nell'ambito dei social network il profilo utente è un insieme di informazioni note, inserite direttamente dall'utente stesso.

- **Microblogging**
  Si riferisce alla possibilità di pubblicare brevi messaggi di testo, immagini o altri contenuti multimediali sul proprio profilo o su quello di altri utenti. Può trattarsi di semplici notizie, messaggi di stato o, frequentemente, di riposte alla pubblicazione di un messaggio da parte di altri utenti. Generalmente sono a carattere pubblico, e quindi visibili da tutti gli utenti del network, anche se ancora una volta rivestono primaria importanza le policy di privacy dei singoli soggetti.

- **Messaggistica**
  A differenza del caso precedente, si vuole intendere una forma di comunicazione tipicamente privata, ristretta ad una selezione di utenti destinatari (al limite uno solo). Spesso è disponibile anche in forma *live* attraverso un meccanismo simile all'instant messaging, ed è

frequentemente impiegata per attuare forme di *live meeting* e *live cooperation*.

- **Gruppi di utenti e micro-comunità**
  Da porre in evidenza anche la possibilità di gestire sottoinsiemi di utenti con interessi o affiliazioni in comune. Interessante come sia generalmente attuare tutte le funzionalità precedenti senza necessariamente distinguere singoli utenti da gruppi di essi, con conseguente flessibilità e facilità d'uso (ad esempio, l'invio di messaggi o la pubblicazione di notizie).

- **Scambio di documenti**
  Per quanto non propriamente intesa in questa accezione, la condivisione di documenti o file è un'altra caratteristica distintiva. Una casistica molto comune riguarda la condivisione di foto e immagini, spesso accompagnata da funzioni accessorie quali il *tagging* o la geolocalizzazione. In altri casi, invece, sono offerti strumenti più completi, adattati alla redazione di documenti, fogli di calcolo o presentazioni, come Google Documents.

5.2 Facebook

Il network certamente più famoso e visitato al mondo, Facebook rappresenta anche, con tutta probabilità, il prodotto funzionalmente più ricco e sofisticato: è integrato in moltissime applicazioni e siti web esterni, disponendo di una vasta API che permette l'impiego di ogni funzionalità in software di terze parti. Il gran numero di operazioni disponibili e loro varianti rendono la trattazione molto complessa, ragion per cui saranno solo accennate con una breve introduzione.

5.2.1 Amicizie e gruppi

L'aspetto centrale del network sono le reti di "amicizia" tra gli utenti: la procedura per stabile un legame tra due utenti prevede che il primo invii una richiesta d'amicizia, a cui il secondo dovrà rispondere in modo affermativo. Volendo, gli amici possono anche essere organizzati in liste, per una migliore gestione dei contatti.

Figura 6 Profilo Facebook del Politecnico di Milano

Una variante di questa meccanica consiste nell'adesione ai gruppi, tipicamente aperti a tutti gli interessati e che solo raramente richiedono un'approvazione da parte di un utente amministratore per quel gruppo. Infine, esistono le pagine, che possono rappresentare argomenti d'interesse comune, personalità pubbliche e/o storiche, eventi, luoghi o istituzioni (come mostrato in figura). In quest'ultimo caso il funzionamento è semplificato, dove gli utenti possono semplicemente esprimere il proprio interesse con il noto pulsante "Mi Piace".

Punto centrale, in ogni caso, è che i messaggi pubblicati sulle bacheche degli utenti collegati saranno visibile anche all'utente, tramite una visualizzazione aggregata nella propria pagina personale (da non confondere assolutamente con il profilo personale).

5.2.2 Messaggistica e commenti

Stabilita una rete di contatti, Facebook permette lo scambio di messaggi in diverse forme: la più semplice è certamente la pubblicazione direttamente sul profilo utente del destinatario (più propriamente la bacheca), il che però rende l'intera conversazione pubblica, almeno entro la cerchia delle rispettive amicizie (l'effetto può però variare in base alle impostazioni della privacy). È però

importante osservare come spesso gli utenti pubblichino messaggi sulla propria bacheca, nell'accezione di una frase o un commento personale. Alternativamente, invece, è possibile inviare messaggi privatamente ad uno o più destinatari, che saranno i soli in grado di visionare l'intera discussione.

Una diversa forma di messaggi è invece rappresentata dai commenti, applicabile praticamente ad ogni oggetto pubblicato sul network: messaggi, immagini ed quant'altro. Spesso le conversazioni nascono da un *post* sulla bacheca di un utente, che dà il via ad ulteriori messaggi da parte degli stessi o, più comunemente, altri utenti "amici". Tutti questi messaggi sono visibili pubblicamente nel rispetto dei limiti di privacy imposti dagli utenti. Un leggera, ma profondamente significativa variante, sono invece i commenti chiusi binari, più comunemente noti come Mi Piace/Non mi piace (Like/DontLike in inglese): in questo caso gli utenti non inseriscono un messaggio esteso, ma si limitano ad esprimere il proprio dis/apprezzamento per un certo messaggio, commento, fotografia, etc… La grande innovazione è rappresentata dalla grande immediatezza e semplicità, anche d'interpretazione, di questi commenti, tanto che l'idea è stata prima estesa anche i siti web di terze parti e quindi esportata anche in altri prodotti concorrenti.

5.2.3 Album fotografici

Un altro dei punti chiave di Facebook è rappresentato dalla convidisione di immagini e fotografia, procedimento quanto mai semplificato e immediato, oltre che vastamente supportato, specialmente su dispositivi mobili dotati di fotocamera. È possibile organizzare ogni elemento in album, ma l'aspetto più noto ed utilizzato e il cosiddetto *phototagging*, che permette di inserire piccoli frammenti visivi in corrispondenza di volti o soggetti ed associarli ad altri utenti. A loro volta questi potranno accedere anche a materiale fotografico non in loro possesso, ma comunque associabile tramite i tag, per mezzo della propria bacheca. L'importanza dell'aspetto sociale in quest'ambito è massima, dal momento che le immagine di un utente non sono pubbliche esclusivamente dal soggetto in questione anche da altri membri; purtroppo sono stati accertati anche utilizzi dannosi di questa funzionalità, spesso oggetto di critiche e di difficile interpretazione da parte della legislatura.

5.2.4 Applicazioni e giochi

Ultima caratteristica presa in considerazione in questa breve esposizione sono le applicazioni e i giochi sviluppati da terze parti tramite le API fornite ufficialmente dalla piattaforma. Il grande vantaggio in questo caso è la

possibilità di coinvolgere altri utenti mostrando loro i risultati e i punteggi conseguiti pubblicandoli sulla propria bacheca personale, oltre ad organizzare facilmente squadre o tornei tramite gli strumenti disponibili sul network stesso.

5.3 Twitter

Twitter è un servizio gratuito di rete sociale e microblogging che fornisce agli utenti una pagina personale aggiornabile tramite messaggi di testo con una lunghezza massima di 140 caratteri. Gli aggiornamenti possono essere effettuati tramite il sito stesso, via SMS, con programmi di messaggistica istantanea, posta elettronica, oppure tramite varie applicazioni basate sulle API ufficiali. Gli utenti possono creare la propria rete sociale tramite il meccanismo del *following*, sottoscrivendo cioè i tweet pubblicati da altri utenti per ottenere una visualizzazione aggregata degli ultimi messaggi pubblicati, con aggiornamenti praticamente in tempo reale.

Per questo motivo, il servizio è diventato estremamente popolare, anche grazie alla semplicità ed immediatezza di utilizzo. Esistono diversi esempi in cui Twitter è stato usato dagli utenti per diffondere notizie, come strumento di giornalismo partecipativo. Ad esempio, nel caso del terremoto in Abruzzo del 6 aprile 2009, gli utenti Twitter hanno segnalato la notizia prima dei media tradizionali.

Figura 7 Pagina Twitter del Politecnico di Milano

La Figura 7 riporta un profilo Twitter di esempio, dove sono evidenziate alcune aree in particolare: la prima è relativa al nome completo o reale dell'utente, a fianco del proprio *username* Twitter, che identifica poi l'informazione di maggior interesse per coloro che intendono comunicarvi. Il secondo punto d'interesse è invece il *follow button*, che permette di sottoscrivere l'elenco di tweet pubblicati dall'utente, visibile in basso nell'ultima area evidenziata.

A margine dei *tweet* standard, esiste anche la possibilità di avviare discussioni private tra due utenti, sempre però tramite tweet di 140 caratteri. Singolarmente, invece, il servizio non offre nativamente alcuna funzione per condividere fotografie o documenti, demandando questo compito a provider esterni come TwitPic per le immagini.

5.3.1 Hashtags

Una particolarità dei messaggi brevi su Twitter è la possibilità di essere etichettati con uno o più *hashtag*, ovvero parole o frasi precedute dal simbolo cancelletto (#) con più parole concatenate, come ad esempio: "Dal TGR Lombardia del 28/07 servizio sul Compasso d'oro al Politecnico al minuto 13.33 #compassodoro #design bit.ly/olQ3H3" (@polimi, 28/07/2011).

In questo modo è possibile condurre ricerche basate sul termine #design e ogni messaggio etichettato con la parola cercata apparirà nei risultati di ricerca. Questi hashtag appaiono anche in un certo numero di siti web di termini più trattati (trending topics), tra cui la homepage di Twitter.

Gli hashtag di Twitter possono essere utilizzati per seguire una discussione tra più persone, incoraggiandone altre a partecipare. Un fenomeno specifico degli ecosistemi Twitter sono i *micro-meme*, identificabili come questioni emergenti o argomenti particolarmente popolari, ed associate immediatamente ad un preciso hashtag; il tratto più specifico, però, è il fatto che questi siano ampiamente usati solo alcuni giorni, per poi cadere rapidamente in disuso. L'immagine sottostante, ad esempio, riporta gli hashtag più popolari del 04/09/11, posti in evidenza sulla pagina principale (se l'utente è registrato).

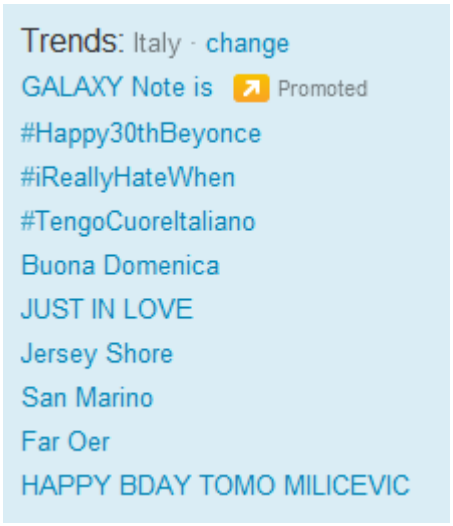


Figura 8 Elenco degli hashtag più popolari al 4 settembre

5.4 Google+

Google + è un social network gratuito fondato da Google. Il servizio è stato lanciato il 28 giugno 2011, in fase test e solo su invito. Il progetto si presenta come rivale di Facebook con alcune varianti funzionali.

La prima differenza significativa è la possibilità di poter suddividere i contatti tramite "Circle" (cerchia), decidendo tra amici, conoscenti, lavoro, famiglia con la possibilità di creare altre categorie, puntando ad un buon livello di privacy. Agendo sulle impostazioni è quindi possibile limitare la diffusione dei dati personali, di qualsiasi notizia o pubblicazione, alle varie cerchie. In Figura 9 è mostrato l'aspetto della pagina di gestione delle cerchie: da un lato è presente l'elenco dei contatti, mentre dall'altro una griglia con le cerchie vere e proprie.

L'utente può trascinare un contatto in una o più cerchie per creare la propria organizzazione personale. In Figura 10, invece, è mostrato come all'atto della pubblicazione di un messaggio sul proprio profilo sia poi possibile utilizzare le cerchie prima definite per limitare la visibilità dei contenuti pubblicati.

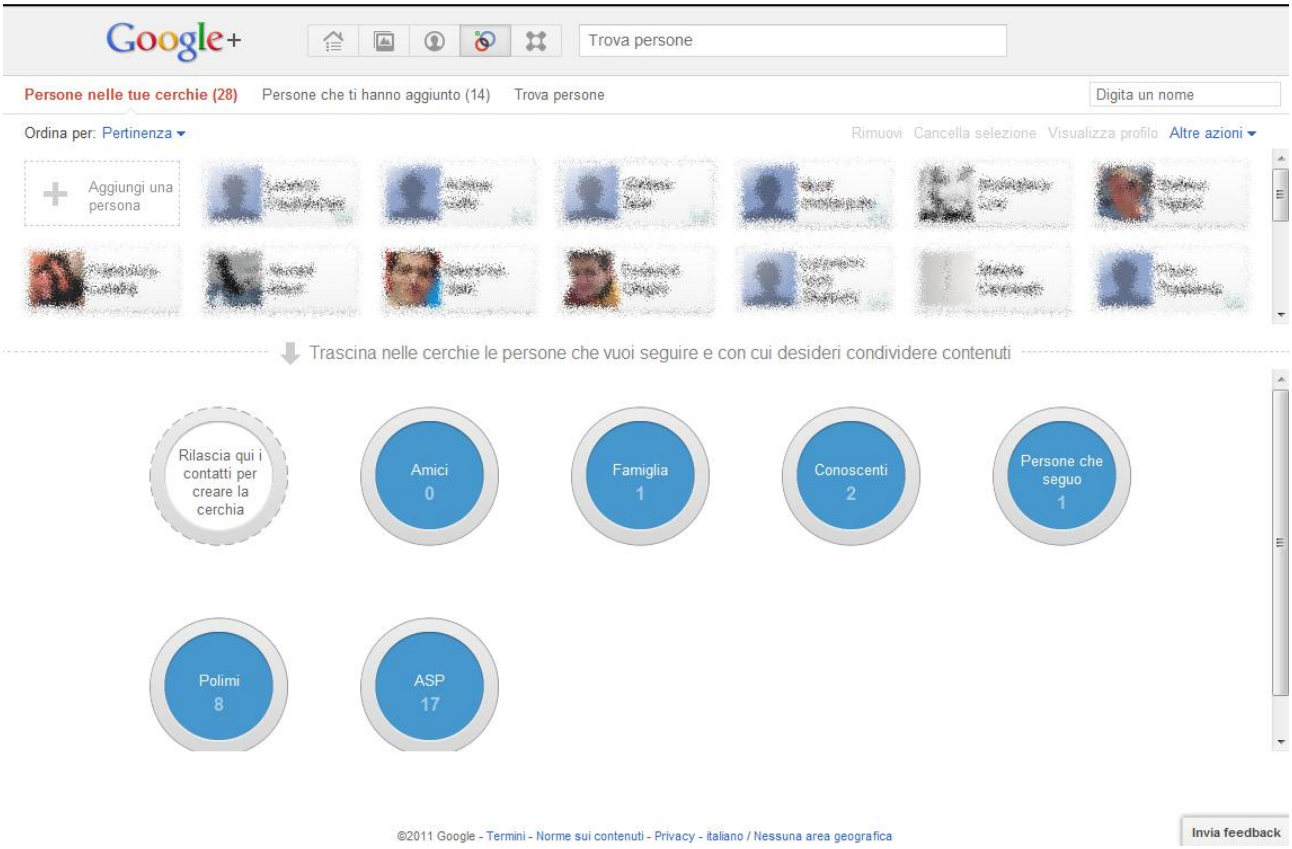


Figura 9 Esempi di cerchie in Google+

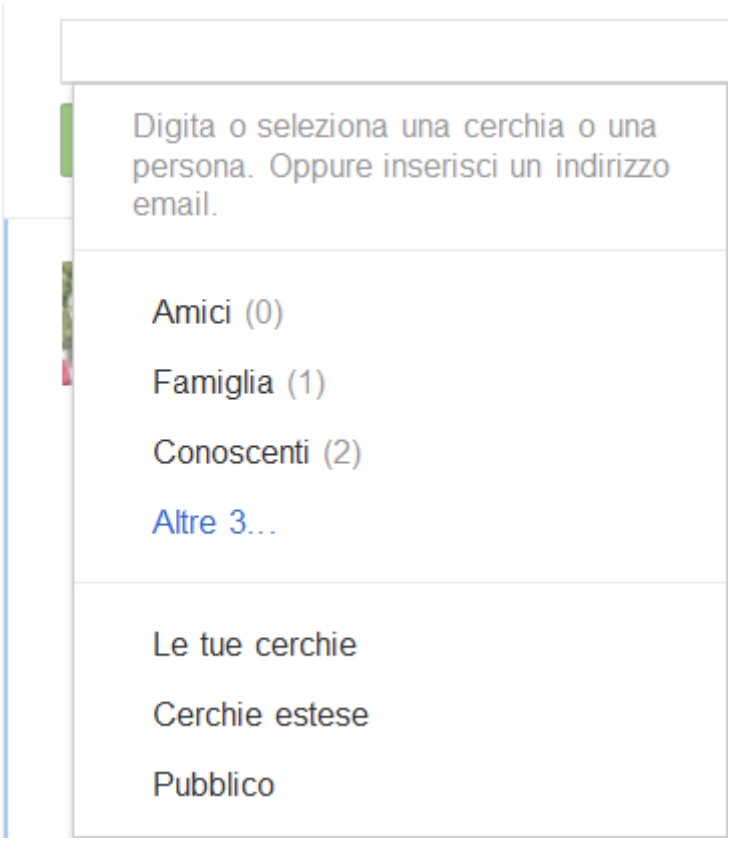


Figura 10 Utilizzo delle cerchie nella publicazione di messaggi in Google+

Altre novità di rilievo, rispetto ad altri social network più affermati, riguardano l'introduzione di nuovi contenuti multimediali, come la possibilità di avviare sessioni audio, video ed entrare a far parte di "video ritrovi", stanze virtuali che permettono di avviare sessioni di *videoconferencing*. Sempre tramite la chat gli utenti hanno la possibilità di scambiarsi file e documenti.

Un'altra interessante funzione di Google+ è detta "sparks", in italiano "spunti", che consente di creare feed semplicemente dopo aver inserito in un box l'argomento interessato. Si creerà quindi uno stream di contenuti inerenti all'argomento scelto, che saranno poi condivisibili con i propri amici.

5.5 Youtube

Pur non essendo considerato un social network puro, la struttura di questo famoso prodotto per la condivisione di video (oggi nell'orbita Google) presenta molte delle funzioni tipiche: innanzitutto, ogni utente registrato dispone di un proprio profilo, detto canale, che funge da punto di raccolta dei propri contenuti caricati, e il cui aspetto può anche essere personalizzato a seconda dei gusti dell'utenti.

Pur non essendo presente propriamente un rapporto simmetrico di connessione tra utenti, è comunque possibile sottoscrivere un canale per ricevere aggiornamenti sulla pubblicazione di nuovi elementi. Ciascun video, poi, può essere commentato da ogni altro utente, purché registrato, tramite un messaggio scritto, o, in alternativa, tramite un meccanismo del tutto identico al Mi Piace introdotto da Facebook.

Com'è noto, infine, è sia possibile incorporare i video caricati dagli utenti in altri website tramite il player omonimo, sia accedere in via programmatica attraverso le API ufficiali per integrarne le funzionalità in applicazioni di terze parti.

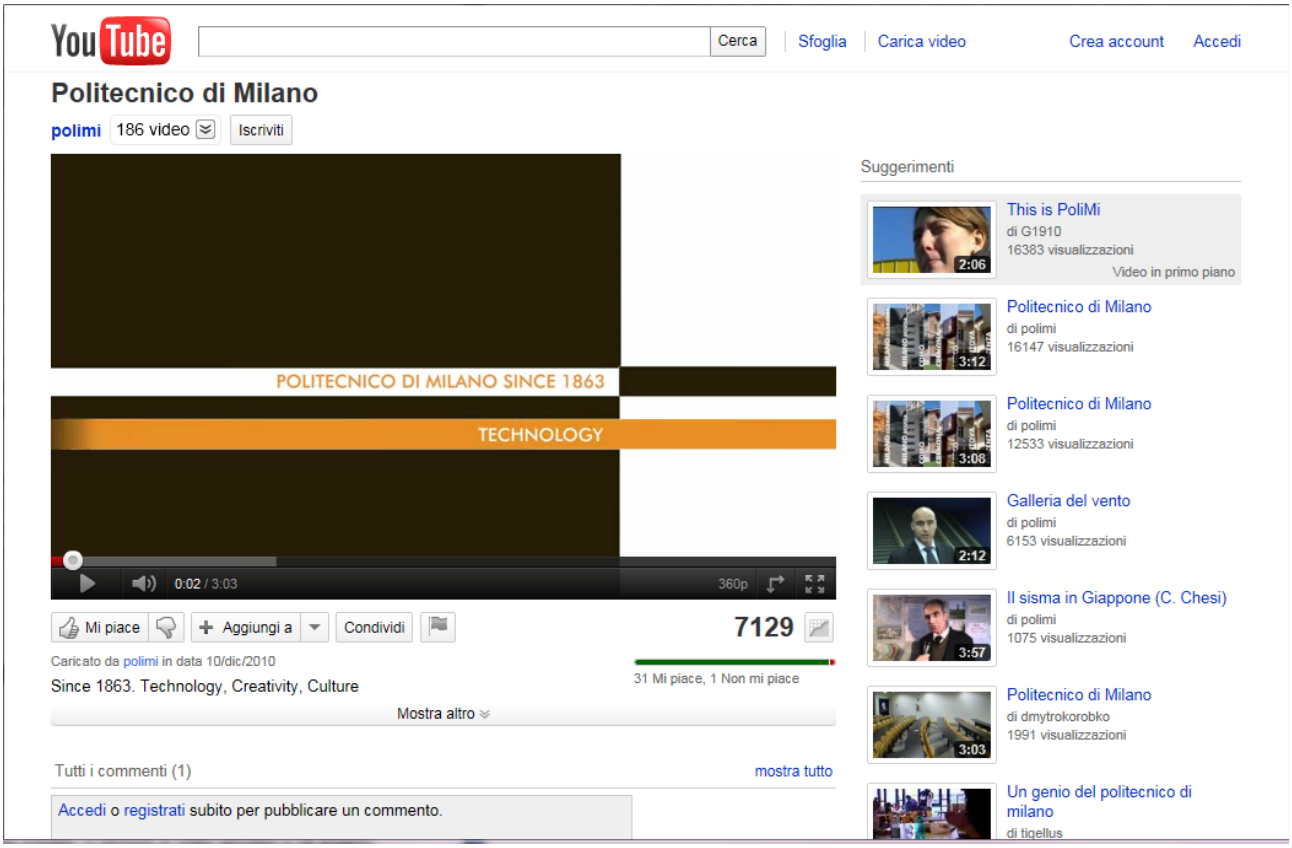


Figura 11 Un video pubblicato dal Politecnico di Milano

5.6 Slideshare

Prodotto molto simile a YouTube, ma focalizzato sulla condivisione si *slide*, nei formati più diffusi come Microsoft PowerPoint, Apple KeyNote, OpenOffice o PDF. Permette la condivisione aperta o limitata dei contenuti caricati, così come un meccanismo per l'incorporamento in applicazioni e siti web esterni. Gli utenti registrati possono poi lasciare i propri commenti e suggerimenti sul materiale visionato.

Figura 12 Esempio di presentazione su SlideShare

La Figura 12 presenta un esempio di presentazione condivisa su SlideShare. In particolare, sono evidenziate alcune aree: la prima, a sinistra, è relativa alla possibilità di condividere il contenuto in oggetto (o quantomeno un link) per mezzo di altri network e siti di terze parti. La seconda area, a destra, propone invece un aspetto interessante, quello del *follow* di un utente, in modo del tutto simile a quanto visto per Twitter. In ultimo, in fondo, è presente il box per i commenti di altri utenti registrati, che possono così lasciare la propria opinione o feedback relativa alla presentazione appena vista.

5.7 Yammer

Rispetto a tutte le altre realtà fin qui descritte, Yammer si differenzia proponendo una diversa gestione del proprio bacino d'utenza: invece di considerare tutti gli utenti registrati come paritari, questi vengono automaticamente associati alla propria organizzazione per mezzo dell'indirizzo email aziendale o istituzionale fornito in fase di registrazione. Per questa ragione, Yammer aspira proprio ad una utenza di tipo aziendale, garantendo un

maggior supporto tecnico, oltre ad una vasta gamma di API, indispensabile per integrare il proprio network all'interno della infrastruttura IT dell'organizzazione.

Resta, tuttavia, un servizio gestito remotamente, così come le informazioni e i dati degli utenti.

5.8 Limitazioni dei servizi pubblici

Generalmente i social network pubblici sono soggetti a critiche abbastanza uniformi, un aspetto in particolare merita attenzione: **il controllo delle informazioni**. Nella fattispecie il problema è piuttosto articolato ed è opportuno analizzarlo con maggior dettaglio: un primo aspetto coinvolge la gestione diretta dei dati da parte del provider. In assenza di una tecnologia adatta, non è infatti possibile avere garanzie circa eventuali accessi non autorizzate alle proprie informazioni, archiviate esclusivamente presso i data center del provider.

Un diverso problema riguarda invece l'accesso da parte di terzi: un primo caso è rappresentato da partner o aziende esterne interessate a svolgere operazioni di *data mining*, vista la grande mole di dati a disposizione del servizio. Secondariamente, anche la gestione delle notifiche da parte di siti web affiliati desta preoccupazione: negli scorsi mesi è divenuto famoso il caso del "like button" di Facebook, per il quale è stata dimostrato che, nel caso l'utente non abbia effettuato il logout dal social network, la presenza del pulsante "Mi piace" all'interno è sufficiente affinché venga tenuta traccia della visita al sito, senza necessariamente richiedere la pressione del pulsante stesso.[10] Un ulteriore punto di criticità, poi, è **l'integrazione** tra un social network e altri sistemi informativi. Nonostante siano spesso disponibili interfacce remote per l'accesso al servizio, infatti, non sempre consentono il pieno controllo delle funzionalità e talvolta adottano protocolli e tecnologie difficilmente adattabili ad esigenze diverse.

[10] http://blogs.msdn.com/b/james_brown/archive/2010/12/07/gov2-0-and-facebook-like-buttons.aspx

Parte II
Soluzione concettuale

6. Meta modello di processo

6.1 Introduzione

In questo capitolo viene descritto un modello per la definizione di processi, in modo da fornire una base per il seguito della trattazione e i successivi sviluppi e modifiche proposti in relazione ai concetti di flessibilità e coinvolgimento sociale. Ciononostante, l'obiettivo del formalismo qui proposto non è fornire un'accurata e completa serie di strumenti per gestire la definizione di qualsiasi processo, quanto piuttosto, individuare gli aspetti più rilevanti e significativi per il contesto in esame, che focalizza la propria attenzione su particolari tipologie di processo. In tal senso, quindi, è più opportuno parlare di *meta modello* di processo, enfatizzando come i concetti descritti siano intensi ad un livello di astrazione superiore rispetto ad altri linguaggi o schemi per la definizione di processi, ai quali è possibile ricondursi una volta stabilitone l'utilizzo di uno scenario reale.

6.2 Attività

Nel presente contesto, un'attività è intesa come la tupla <N, I, O>:

- **(N)ame:** nome dell'attività, coincidente con il nome del metodo
- **(I)nputModel:** la struttura dati richiesta come input dal metodo, tipicamente i parametri formali opportunamente tipizzati.
- **(O)utputModel:** la struttura del risultato dell'invocazione del metodo.

In merito alla definizione dei tipi di dato è possibile equipararne il potere espressivo, con buona approssimazione, alla specifica XML Schema: sono quindi previsti sia tipi elementari, come stringhe, interi, booleani, etc… sia strutture complesse ottenute per aggregazione degli stessi, unitamente a costrutti di ripetizione (tipicamente liste). Onde evitare un'eccessiva complessità, però, costrutti specifici come le restrizioni sul formato delle stringhe o le limitazioni sul range dei valori numerici non sono stati considerati.

Secondariamente è bene sottolineare come un'attività così definita non sia un'entità attiva, quanto piuttosto una specifica comportamentale, da distinguersi quindi rispetto all'**esecuzione (o istanza) della stessa, denominati task**. Ciascuno di questi ultimi è definibile come una tupla <L, A, D, S,U>:

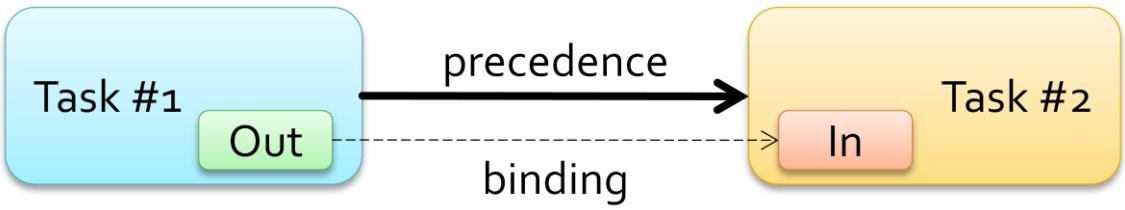
- **(L)abel**: un'etichetta descrittiva dell'istanza attuale dell'attività

- **(A)ctivity:** l'attività (concettuale) di riferimento
- **(D)ata:** i dati di input specifici per questa esecuzione forniti al metodo del controller. È interessante osservare fin d'ora che del tutto ammissibile inserire "manualmente", all'atto della istanziazione dell'attività, solo una parte delle informazioni richieste dall'InputModel. Le mancanti potranno infatti essere completate automaticamente dal sistema sulla base dell'OutputModel di una attività precedentemente completata (vedi segg.)
- **(S)tate:** lo stato attuale dell'esecuzione (vedi segg.)

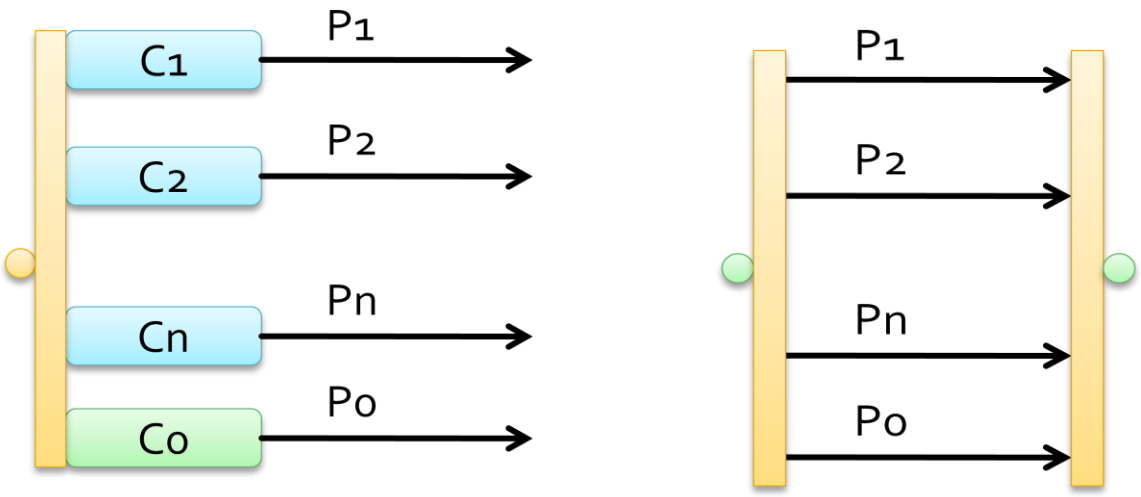
6.3 Costrutti o attività speciali

In prima istanza, diverse activity possono essere collegati tra loro in modo sequenziale e ciò permette di creare un'*associazione*, denominata **binding**, tra i rispettivi output e input, in modo tale che all'avvio dello i+1-esima activity i dati siano degli step precedenti siano già consolidati. È però necessario sottolineare che una relazione di precedenza tra activity non implichi necessariamente l'obbligo a definire *binding* tra output e input. È definita quindi la funzione *Bind(S,T,B)*, dove:

- (S)ource, rappresenta l'attività di origine
- (T)arget, è l'attività obiettivo
- (B)indings = $\{(p_i, p_j)\}$ è l'insieme dei *binding*, intesi come coppie (p_i, p_j), dove p_i una proprietà dell'OutputModel di Source e p_j una proprietà dell'InputModel di Target.



In secondo luogo, è possibile impiegare dei costrutti più complessi per variare il flusso di esecuzione del processo: **choice** e **parallel**. Il primo definisce una serie di percorsi alternativi P_i, ciascuno accessibile in presenza di un data condizione d'ingresso C_i. In aggiunta, si ipotizza la presenza in un percorso di *default* P_0 eseguito nel caso nessuna delle condizioni d'ingresso sia verificata (in altre parole la condizione C_0 è sempre verificata).



Al contrario, il costrutto *parallel* consente di eseguire più percorsi in contemporanea, in modo indipendente gli uni dagli altri. In questo caso, però, è necessario un costrutto complementare, **barrier**, che blocchi l'esecuzione del flusso fino al completamento di tutti rami paralleli.

In ultimo, sono previsti anche i concetti di *Start* ed *End*, con le seguenti, e intuitive definizioni:

- Per ogni processo, esistono esattamente un costrutto *Start* ed esattamente un costrutto *End*
- Il costrutto *Start* non ammette alcuna dipendenza
- Per ogni processo, non esiste alcuna attività che dipenda dal costrutto *End*.
- Entrambi non sono associati ad alcun controller e si comportano come un'activity vuota: pertanto, sia l'InputModel sia l'OutputModel sono vuoti.

6.4 Risorse

Le risorse sono in effetti un concetto assiomatico, ma nell'accezione qui presentata si intendono in modo particolare **solo risorse condivise tra le diverse attività di un intero processo**. Non si definiscono risorse, pertanto, gli output intermedi prodotti dall'avvenuta esecuzione delle attività, collegabili poi ad una successiva per mezzo del meccanismo di binding prima descritto.

In questa fase non viene proposta una descrizione più precisa della possibile struttura di una risorsa, in quanto ciò comporterebbe la necessità di individuare un particolare contesto di riferimento, vanificando, anche se parzialmente, la generalità della trattazione fin qui adottata. Viceversa, è invece possibile individuare alcune proprietà formali d'interesse, ancorché generali,

come l'impostazione di sola lettura: pur non precisando la particolare natura delle risorse coinvolte nei processi, la distinzione tra risorse modificabili e non modificabili è certamente rilevante e comunque sufficientemente generale. Formalmente, si definisce l'insieme delle risorse non modificabili come segue:

$$NotModifiable \subseteq R$$

Dove R è l'insieme di tutte le risorse attinenti il processo (si veda il paragrafo seguente). Tale proprietà rivestirà un significato estremamente importante nel seguito. La condivisione delle risorse, infatti, introduce un potenziale ulteriore elemento di criticità, nella forma delle ben note *race conditions*: il presente formalismo permette, tramite il costrutto di split, di gestire percorsi multipli paralleli e pertanto sarebbe del tutto ipotizzabile il verificarsi di accessi simultanei non gestiti alla medesima risorsa.

Tale considerazione avrebbe effetti potenzialmente profondi sulla gestione dei processi, richiedendo una formulazione e un metodo del tutto simile all'accezione di processo nei sistemi operativi, ad esempio. Nell'intendo di mantenere un approccio semplice, si è scelto invece di prevenire ogni potenziale race condition per costruzione: agendo nella fase di creazione di un processo, si vuole inserire un vincolo che impedisca di definire blocchi split in cui almeno un'attività modifica una risorsa:

$$NoRaceConditions(P) \leftrightarrow \nexists s \in P \wedge Split(s) | \exists t \in s | \exists r \notin NotModifiable | Writes(t, r)$$

$$Writes \subseteq A \times R$$

Dove la relazione *Writes* rappresenta il fatto che una data attività modifichi la risorsa r, posto logicamente detta risorsa sia modificabile. Con questa formulazione sono di fatto impedite tutte le scritture in qualsiasi blocco split, fatto che potrebbe suscitare perplessità circa le limitazioni introdotte. Tuttavia, grazie proprio alla generalità del modello di processo, in particolare in riferimento alle attività, è possibile configurare una vasta gamma di soluzioni alternative: ad esempio, ipotizzando che le risorse assumano l'accezione di documenti, un'attività prima del blocco split potrebbe richiamare un software di versioning e quindi una successiva applicare un merging delle diverse copie.

6.5 Utenti e attori

La definizione di utenti deve considerare la problematica di specificare gli attori coinvolti nel processo, e in particolare l'assegnazione delle attività, oltre che soddisfare i tipici requisiti di controllo dei permessi. In virtù di ciò, si definisce:

$$U = S \cup G \cup C$$

Dove S è l'insieme degli utenti singoli o semplici. Non volendo in questa sede essere eccessivamente restrittivi e specifici, onde mantenere la presente formulazione sufficientemente generale, si intenderà il singolo utente come un concetto assiomatico, senza ulteriore descrizione. G rappresenta invece l'insieme dei gruppi di utenti (semplici):

$$G = \{g_i\}, g_i = \{u_1 \cdots u_n\}, u_i \in S$$

Si noti come la definizione non permetta la presenza di gruppi eterogenei, ossia composti a loro volta da altri gruppi: per quanto potenzialmente utile per rappresentare alcune gerarchie, tipicamente diverse realtà aziendali, tale scelta comporterebbe la necessità di garantire controlli di consistenza onde evitare ciclicità nelle relazioni di appartenenza di utenti, in senso esteso, a gruppi.

Il significato dell'insieme C, invece, corrisponde al concetto delle classi di utenti. Analogamente ai gruppi, le classi rappresentano aggregazioni di utenti semplici, ma non supportano operazioni di join e leave. Si definiscono cioè come una partizione dell'insieme S, tale per cui ogni utente appartiene a esattamente una sola classe:

$$belongsToClass: S \rightarrow C$$

$$\forall u \in S | \exists c \in C \, | belongsToClass(u, c) \rightarrow \nexists \bar{c} \in C | belongsToClass(u, \bar{c}) \wedge c = \bar{c}$$

Il concetto di classe riveste un certo significato da un punto di vista teorico per le sue peculiarità ed è un utile strumento concettuale per la modellazione di determinati contesti applicativi.

Fornita una specifica per gli attori, è necessario definire una forma per la gestione dei permessi e degli assegnamenti attività – attore. S'introduce così l'insieme ACL (*Access Control List*):

$$ACL = \{\langle a_i, p_i \rangle\},$$

$$p_i = \langle u, c \rangle,$$

$$a_i \in A, u \in U, c \in \{run, edit\}$$

Intuitivamente, l'insieme contiene coppia attività – permesso, a propria volta identificato dall'utente coinvolto (in senso ampio, secondo la definizione dell'insieme U) e dalla specifico *constraint* (vincolo) imposto. Sono *constraint* validi l'esecuzione di un'attività e la sua modifica.

6.6 Processo

In ultimo, un processo è rappresentabile come una tupla P = <A, D, U,R>, con il seguente significato:

- **(A)ctivities:** l'insieme delle attività inserite nel processo, **incluse le attività speciali**.
- **(D)ependencies = {<a_i, a_j, B_{ij}>}:** l'insieme delle relazioni di precedenza tra le attività. Ciascuna è definita come la terna <a_i, a_j, B_{ij}> dove a_i rappresenta l'attività di riferimento e a_j la dipendenza, con B_{ij} inteso come il set di *binding* tra le due.
- **(U)sers:** qualsiasi attore del sistema, interessato al processo in oggetto.
- **(R)esources:** le risorse coinvolte nel processo

La scelta di riunire in un unico set le attività e i costrutti di controllo permette di assimilare l'intero processo ad un grafo (almeno in prima approssimazione). Data questa osservazione, un processo è **valido** se è descrivibile come un DAG (Directed Acyclic Graph).

6.7 Stato delle attività

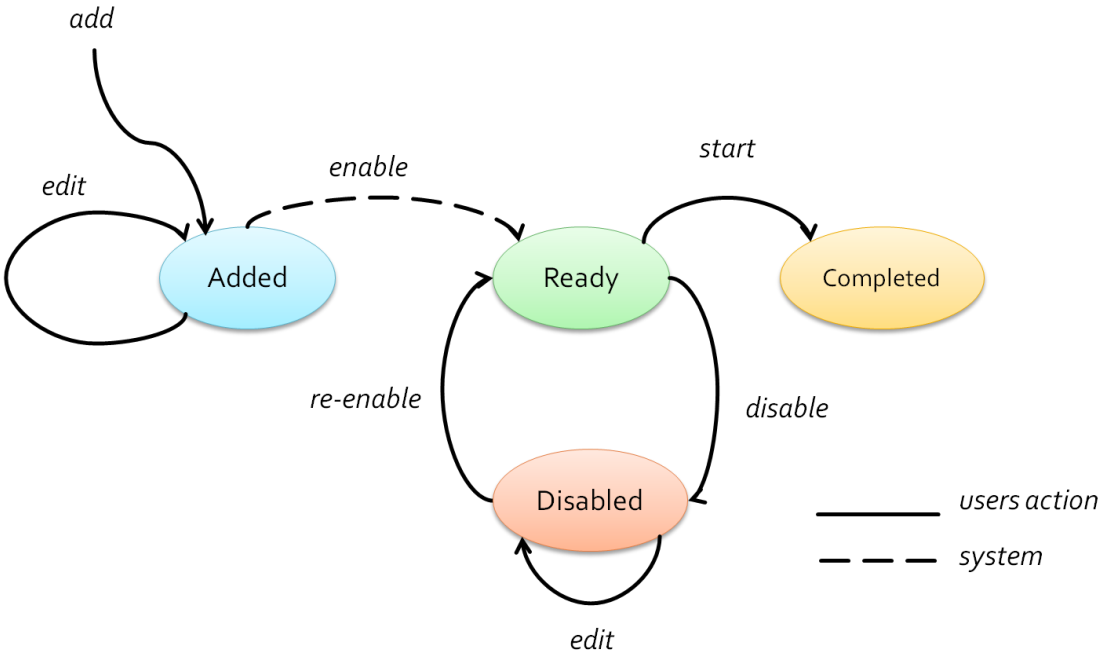


Figura 13 Schema degli stati di un'attività

È stato menzionato in precedenza come i processi siano concepiti in modo da essere flessibili, e quindi modificabili anche ad esecuzioni già iniziata. Ciò si deve alla particolare concezione dello stato delle diverse attività: come mostrato nel diagramma qui riportato, una volta aggiunta, un'attività può essere **continuamente modificata dall'utente**. In questo stato (*added*), l'activity fa parte del processo, ma non può ancora essere eseguito a causa delle **dipendenze da altre activity precedenti**. In questo caso, e soltanto per questo stato, **il sistema**, e non l'utente, deve intervenire per porre il task in stato di *ready* una volta soddisfatte tutte le dipendenze. Tale scelta è volta a prevenire incongruenze nei binding tra attività, che non sarebbero attuabili, ad esempio, se l'output del task i-1-esimo non fosse ancora disponibile al momento dell'attivazione del task successivo. Raggiunto lo stato di *ready*, gli utenti possono ancora applicare cambiamenti al task, a patto però di **disattivarlo**. Ancora una volta, infatti, è necessario prevenire la situazione in cui l'utente amministratore stia modificando il task ed un qualsiasi utente sia invece prossimo ad avviarlo.

Esiste però un altro aspetto da considerare circa lo stato di un'attività, poco evidente ma decisamente rilevante per il corretto svolgimento di un processo. Si ipotizzi che un certa attività sia assegnata non ad un utente singolo, ma ad un gruppo o classe. In questo caso, non esiste un concetto di stato in

assoluto, ma di stato di un'attività per un dato utente. Formalmente è necessario definire l'oggetto *State* come segue:

$$State = \{\langle p, a, u, s \rangle\}$$

$$p \in P, a \in A, u \in U, s \in ActivityStates$$

Ciascuna tupla permette di definire lo stato di una determinata attività, per un certo utente e processo, così da permettere una corretta gestione della transizione degli stati per ogni attore coinvolto.

La questione tuttavia non è del tutto risolta: si ipotizzi un semplice processo composto di due attività, A e B, con B che dipende da A. L'attività A, tuttavia, è assegnata a più attori, come nell'esempio precedente. In questo scenario nasce una problematica circa la risoluzione della dipendenza di B da A: quando B può essere avviata? È possibile definire due casi: quando tutti gli attori di A hanno svolto tale attività, oppure quando almeno uno di essi ha completato l'assegnamento. Per entrambi i casi, comunque, la soluzione è introdurre un ultimo costrutto:

$$Counter = \{\langle p, a, c \rangle\}$$

$$p \in P, a \in A, c \in \mathcal{N}$$

Molto semplicemente viene mantenuto un contatore del numero di utenti che hanno completato un'attività, e quindi, nel primo caso, la dipendenza viene risolta nel momento in cui tutti gli attori coinvolti da A ne hanno completato l'esecuzione; nel secondo, invece, è sufficiente che una sola esecuzione.

7. Dimensione flessibilità

7.1 Introduzione

Terminata la definizione del modello formale, la trattazione prosegue con l'introduzione del concetto di flessibilità. Più precisamente, tale aspetto concerne le singole attività ed è inteso come la possibilità di legarne l'effettiva esecuzione ad una o più funzionalità fornite da un servizio remoto (tipicamente un servizio web o una serie di API).

Ad esempio, si supponga di volere programmare un meeting e di dover concordare con i partecipanti la data e l'orario. È possibile definire facilmente questo processo piuttosto semplicemente, ma come realizzarlo concretamente? Una possibilità potrebbe consistere nella implementazione *ad hoc* delle due funzioni principali richieste, raccolta delle disponibilità e programmazione del meeting. Questa soluzione, tuttavia, presenta molti lati problematici e pochi vantaggi: innanzitutto occorre analizzare ogni task da svolgere e considerare almeno un'analisi dei requisiti essenziale, in modo da acquisire una conoscenza di base del contesto. Secondariamente è necessario impiegare risorse, specialmente temporali, per la realizzazione del software adatto, il che, in caso di attività e processi più sofisticati, potrebbe presentare non poche sfide. Ma soprattutto, il prodotto così ottenuto difficilmente sarà completo, nel rispetto del contesto considerato, o integrabile con altre soluzioni esistenti sul mercato: ad esempio, la mancanza di un supporto a dispositivi mobile (ad oggi praticamente imprescindibile) o l'impossibilità di sincronizzare i meeting così programmati nel proprio calendario personale, perché memorizzato da un diverso provider.

7.2 Vantaggi

In virtù di queste considerazioni, si è deciso di optare per una diversa strategia, che si focalizzi sul riuso e sull'integrazione di servizi già esistenti: nel caso preso in esame, si immagini di poter utilizzare tre noti servizi web, Facebook, Google Calendar e Doodle. Per la prima fase, la raccolta di conferme per date e orari del meeting sarebbe possibile procedere in due modi:

1. Impiegare un servizio specifico come Doodle, predisposto essenzialmente per la programmazione di eventi: le funzionalità sono tarate con precisione per lo scenario in oggetto di esame e l'interfaccia utente è semplice e intuitiva. Resterebbe però la necessità di coinvolgere gli utenti

inviando loro delle *invitation mail* (o quantomeno lo URL del sondaggio una volta pubblicato), cosa che Doodle non gestisce in modo del tutto automatico.

2. Ricorre al noto social network Facebook, pubblicando un post sulla propria bacheca con il messaggio per i propri colleghi e una serie di commenti con le scelte disponibili. In questo modo gli altri utenti coinvolti riceveranno la notifica di pubblicazione e sarà loro sufficiente utilizzare la funzionalità Like/DontLike per esprimere la propria preferenza. Si tratta di un metodo insolito, ma prevede il grande vantaggio di essere immediatamente visibile e a disposizione di ogni utente, anche su dispositivi mobile.

Per quanto concerne invece la programmazione del meeting vera e propria, sarebbe ancora possibile sfruttare una delle funzionalità offerte da Facebook, oppure impiegare il servizio Calendar di Google. Visto lo scenario, quest'ultima potrebbe rivelarsi la scelta migliore, specialmente per la capacità di sincronizzazione dei calendari anche con altri prodotti e applicazioni esistenti.

In ultima analisi, anche il cloud computing, in particolare attraverso il paradigma Software-as-a-Service, può fornire ulteriori prospettive in questa direzione. È lecito ipotizzare, infatti, che in futuro il numero di servizi disponibili per l'integrazione aumenti, molti dei quali esposti tramite sistemi SaaS, in grado di garantire maggiore affidabilità e supportare grandi volumi di traffico.

7.3 Criticità

A margine di questa introduzione, è già possibile individuare tre aspetti particolari nell'approccio qui presentato: innanzitutto è evidente come sfruttando servizi già esistenti sia possibile completare i processi definiti in modo piuttosto semplice; in secondo luogo, l'integrazione con altri prodotti a loro volta integrabili (tramite API, o altre tecnologie) estende transitivamente l'integrabilità della propria soluzione: nell'esempio precedente, pubblicando eventi su Google Calendar, diviene automaticamente possibile accedere a questi dati attraverso altri servizi o applicazioni in grado di comunicare con i prodotti Google (lo stesso dicasi, logicamente, per Facebook). Ultimo ma non meno importante, risulta evidente non solo come diversi provider offrano servizi adatti a gestire lo stesso task, come nel caso in esame, ma anche che due servizi

compatibili con gli stessi requisiti di processo possano svolgere in modo profondamente diversi l'esecuzione di un task. A ciò occorre anche aggiungere come l'impiego di una vasta di gamma di servizi esterni produca una frammentazione delle informazioni, non più persistite ed accessibili in un unico punto d'ingresso.

7.4 Riformulazione

Si vuole ora proporre una soluzione concettuale onde far fronte alle criticità appena illustrate. Innanzitutto è necessario estendere la definizione di un'attività di processo in modo da rendere possibile l'associazione dell'esecuzione di quest'ultima con un particolare servizio. L'obiettivo è mantenere la definizione formale descritta in precedenza e introdurre una nuova notazione: si definisce quindi il concetto di *service binding* come la terna <T, S, C>, con il seguente significato:

$$SB = \langle A, S, C \rangle$$

- **(A)ctivity:** il task, o attività, definita all'interno di un processo.
- **(S)ervice:** il riferimento al servizio esterno da associare
- **(C)onfig:** un insieme di coppie chiave – valore che rappresentano informazioni parametriche aggiuntive specifiche per il servizio invocato, con lo scopo di soddisfare vincoli tecnici particolari o necessarie per sfruttare funzionalità speciali. Ne sono un esempio le eventuali credenziali di autenticazione (per vincoli tecnici) o le opzioni di sincronizzazione degli eventi di calendario (in riferimento al servizio Google Calendar).

Dopodiché, si estende la definizione di processo come la tupla a quattro <A,D,U,B> aggiungendo l'informazione B, come l'elenco di service bindings specificati:

$$P = \langle A, D, B, U \rangle,$$

$$B = \{b_i\}, b_i = \langle a, s, c \rangle$$

$$a \in A, s \in SR$$

Ciononostante, questa riformulazione non è ancora sufficiente, in quanto sono ancora mancanti le informazioni tali da permettere l'identificazione delle classi di task concettuali supportate da un servizio registrato al sistema; questa

frase evidenzia però un secondo problema, ossia la necessità di disporre di un insieme che tenga traccia e "descriva" i servizi esterni noti e supportati. Non è infatti ipotizzabile la possibilità di interfacciarsi con un qualsiasi provider arbitrario, in quanto ciò richiederebbe quantomeno un algoritmo automatico e un descrittore standard per ogni servizio. E questo, allo stato attuale, non corrisponde alla realtà. Come conseguenza si prevede la presenza di un componente software in grado di supportare specificamente la registrazione e la comunicazione verso ciascuno dei servizi esterni (maggiori dettagli verranno presentati nei capitoli seguenti).

In merito al punto iniziale, invece, l'identificazione dei task supportati da un dato servizio può essere modellata come un *Service Registry*:

$$SR = \{sr_i = \langle s, t \rangle\}, s \in S, t \in T$$

Dove S è l'insieme dei servizi noti e T l'insieme delle classi di task definite nel sistema. Si tratta perciò di una funzione di *mapping* tra ogni servizio e l'elenco dei task concettuali. In tal modo è possibile accertare anche la correttezza di un processo esteso secondo l'ultima riformulazione verificando i service binding come segue:

$$CorrectBindings(P) \leftrightarrow \forall b = \langle s, t \rangle \in B \mid \exists sr = \langle r, q \rangle \in SR \mid s = r \,\&\&\, t = q$$

$$P = \langle A, D, U, B \rangle$$

7.4.1 Service binding multipli

Come osservazione finale, si noti che le definizioni qui riportate non escludono che una data attività, in un certo processo, possa essere associata a più servizi. Questa possibilità è indicata per garantire la massima flessibilità possibile in fase di esecuzione del processo, in cui l'utente può decidere se avviare l'attività tramite un solo servizio oppure su ognuno di quelli configurati: si pensi ad esempio ad un processo che preveda la pubblicazione di una notizia; in tal caso è piuttosto comune replicare la pubblicazione su diversi canali, come potrebbero essere il proprio profilo Facebook e un canale Twitter.

Vale la pena sottolineare, tuttavia, come il motore di esecuzione dei processi non svolga una scelta automatica in presenza di binding multipli e come invece l'azione decisionali sia affidata all'utente all'atto dell'avvio della attività.

8. Dimensione sociale

8.1 Introduzione

Proposto un formalismo per la definizione dei processi e discussa la dimensione di flessibilità, si vuole ora analizzare la componente sociale. S'intende cioè introdurre la possibilità, da parte degli utenti, di partecipare attivamente alla definizione e all'esecuzione dei processi. Per prima cosa verrà innanzitutto definito un modello formale di riferimento, e quindi sarà proposta un forma di integrazione con i processi fin qui descritti.

8.2 Modello formale

In precedenza, durante la prima definizione dei processi, è stata fatta una breve menzione circa il significato di "utente" e l'intrinseca importanza che essi rivestono nell'ambito dei processi non automatici. In questa sede si riprende in parte il formalismo proposto, arricchendolo per con ulteriori aspetti maggiormente legati al mondo delle reti sociali. Innanzitutto, la totalità degli utenti era così definita:

$$U = S \cup G \cup C$$

Dove S è l'insieme degli utenti singoli o semplici. Non volendo in questa sede essere eccessivamente restrittivi e specifici, onde mantenere la presente formulazione sufficientemente generale, si intenderà il singolo utente come un concetto assiomatico, senza ulteriore descrizione. G rappresenta invece l'insieme dei gruppi di utenti (semplici):

$$G = \{g_i\}, g_i = \{u_1 \cdots u_n\}, u_i \in S$$

Ancora una volta, si noti come la definizione non permetta la presenza di gruppi eterogenei, ossia composti a loro volta da altri gruppi. Si definisce ora la relazione *belongs* tra utenti singoli e gruppi, nel modo seguente:

$$belongs: S \to G$$

Senza l'aggiunta di ulteriori vincoli, è ammessa la possibilità che uno stesso utente appartenga contemporaneamente a più gruppi. Conseguentemente, poi, si introducono anche due funzioni in grado di formalizzare l'azioni di partecipazione e abbandono di un gruppo da parte di un utente:

$$join: S \rightarrow G$$

$$join(u, g)\ sse\ \sim(u \in G)$$

$$leave: S\ \rightarrow G$$

$$leave(u, g)\ sse\ u \in G$$

I due vincoli esprimono due condizioni di applicabilità delle funzioni, con ovvio significato dei simboli: un utente può partecipare solo a gruppi cui già non appartiene, e specularmente per l'abbandono. Non si precisano invece legami tra le due operazioni e la relazione *belongs*: in linea teorica si potrebbe pensare di vincolare l'appartenenza ad un gruppo all'effettuazione dell'operazione di *join* (ad esempio), ma ciò precluderebbe la possibilità di definire policy di join/leave personalizzate: ne sono il caso gruppi di lavoro per i quali è necessaria un'approvazione, oppure determinate divisioni aziendali cui non si può semplicemente notificare un abbandono. In quest'ambito non preciseremo le possibili formalizzazioni per tali policy, compito oltre gli obiettivi di questa trattazione.

Come ultimo punto, è ancora necessario precisare il significato dell'insieme C: si tratta delle classi di utenti. Analogamente ai gruppi, le classi rappresentano aggregazioni di utenti semplici, ma non supportano operazioni di join e leave. Si definiscono cioè come una partizione dell'insieme S, tale per cui ogni utente appartiene a esattamente una sola classe:

$$belongsToClass: S\ \rightarrow C$$

$$\forall u \in S |\ \exists c \in C\ | belongsToClass(u, c) \rightarrow\ \nexists \bar{c} \in C | belongsToClass(u, \bar{c})\ \wedge c = \bar{c}$$

Il concetto di classe riveste un certo significato da un punto di vista teorico per le sue peculiarità ed è un utile strumento concettuale per la modellazione di determinati contesti applicativi.

Definita quindi l'accezione di utente, si corregge la tupla di processo P = <A,D,S,B>, dove S $\subseteq$ U (la notazione è stata leggermente modificata per evitare ambiguità tra gli utenti coinvolti in un processo dall'insieme della totalità degli utenti).

8.3 Connessioni

L'aspetto specifico e cruciale delle reti sociali è rappresentato dalla possibilità di permettere ad ogni utente di stabilire una serie di connessioni tra essi stessi, secondo modalità decise non in modo sistematico, dall'alto secondo un approccio top-down, quanto piuttosto viceversa: sono gli utenti quindi a collegarsi e a non a essere collegati.

Le forme di relazioni tra gli utenti sono molteplici e considerevolmente mutevoli, ma in ultima istanza è possibile definirle in modo sufficientemente intuitivo:

$$link(u, v), u, v \in U$$

$$link(u, v) \; sse \; v \in L(u) \subseteq U$$

Dove L(u) è l'insieme delle connessioni dell'utente u. Vale la pena notare, invece, come una connessione non sia necessariamente simmetrica: non è generalmente assumibile, cioè, che valga il seguente:

$$v \in L(u) \; sse \; u \in L(v),$$

Nel qual caso si tratterà di una *rete sociale simmetrica*. Da un punto di vista analitico, ciò ha effetti profondi sulla descrizione della rete a livello generale: riformulando con diverso formalismo quanto appena presentato, si potrebbe affermare che nel primo caso è possibile definire una relazione d'ordine parziale tra gli utenti, cui sarebbe possibile attribuire un preciso significato semantico (anche se probabilmente non del tutto immediato). Nel secondo caso, tutto questo non sussiste. Ciò non implica, logicamente, che una rete sociale simmetrica rivesta minor importanza o significatività, ma è bene sottolineare come la proprietà formale di simmetria determini conseguenze non banali.

8.4 Messaggi

Fornita la definizione di utenti, si vogliono ora indagare alcune delle funzionalità più comuni rese disponibili da una rete sociale. La prima presa in esame riguarda lo scambio di messaggi tra utenti: nella sua forma più semplice s'intende una comunicazione tra due utenti, chiusa però rispetto al coinvolgimento di ulteriori soggetti; si tratta altresì di uno scambio privato, che è possibile scrivere come segue:

$$send(u, m)$$

$$m = \langle s, p \rangle,$$

$$u, s \in U$$

Dove la funzione *send* rappresenta l'invio del messaggio m all'utente u. Si noti che il messaggio è poi composto di due parti, il mittente, a sua volta un utente, e un contenuto o *payload*. Onde preservare, ancora una volta, la generalità della trattazione, non si vuole precisare ulteriormente la struttura del contenuto. Riveste interesse, piuttosto, sottolineare come questo formalismo consenta anche di gestire destinatari multipli, a patto di effettuare la correzione seguente:

$$u \subseteq U$$

In realtà, all'interno di reti sociali è piuttosto diffusa anche un'altra forma di comunicazione, aperta invece al coinvolgimento di altri utenti: è il caso della pubblicazioni di messaggi all'interno di profili utente, sia quello del mittente, sia quello del destinatario. L'aspetto particolare di questa soluzione è la visibilità tipicamente pubblica del messaggio, che può così essere letto e condiviso anche da utenti originariamente non coinvolti nella discussione, estendendo il bacino d'interesse dell'argomento. Rispetto al formalismo precedente, è allora possibile definire:

$$publish(u, m)$$

$$m = \langle s, p \rangle$$

$$u, s \in U$$

Il messaggio in questione può quindi essere identico nei due casi, ma la semantica della due funzioni è ben distinta. Ulteriore elemento degno di nota è che mentre nel caso di comunicazione aperta è del tutto ammissibile il caso $u = s$, che corrisponderebbe allo scenario di cui un utente pubblichi un messaggio sulla propria bacheca, è di scarso significato che ciò avvenga nel primo e pertanto si asserisce:

$$send(u, m)$$

$$m = \langle s, p \rangle$$

$$u \subseteq U$$

$$s \in U \backslash u$$

La definizione di messaggio potrebbe poi essere ulteriormente ampliata per tenere conto di una conversazione o discussione nel suo complesso e facilitare l'aggregazione dei messaggi pertinenti tracciando il messaggio originale, nel modo qui riportato:

$$m = \langle s, p, o \rangle$$

$$o \in M$$

$$M = \{m\}$$

In tal modo per ogni messaggio ne è noto un secondo inteso come origine della conversazione e questo consente di raggruppare messaggi afferenti la medesima origine. Potenzialmente, sarebbe anche possibile aggiungere anche altre informazioni, sotto forma di proprietà (coppie chiave-valore), come ad esempio data e ora, luogo (*geotagging*) oppure dati sul dispositivo utilizzato per l'invio (computer, mobile, etc..).

8.5 Risorse

La condivisione di risorse è un certamente un aspetto rilevante nei processi di collaborazione tra utenti, ma corrisponde anche ad un servizio tipico di molte reti sociali. Riprendendo la definizione di risorsa fornita nei capitoli precedenti, è possibile definirne la condivisione nel modo seguente:

$$share(u, r)$$

$$u \in U, r \in R$$

In questo modo, però, risulterebbe del tutto impossibile identificare risorse in particolare, in virtù anche del fatto che non ne è ancora stata precisata la struttura e il contenuto. Volendo ancora una volta circoscrivere le specificità di un dato contesto, è comunque possibile introdurre il ben conosciuto metodo dei *tags*, che si rileva sufficientemente semplice, pur garantendo in ogni caso un buon supporto alla catalogazione e alle ricerche. Si estende così la definizione precedente:

$$share(u, r, T)$$

$$u \in U, r \in R, T = \{t\}$$

Dove intuitivamente T rappresenta l'insieme dei tags applicati; un singolo tag è comunemente identificabile come una stringa, ma implementazioni diverse

potrebbero proporre alcune varianti, ragion per cui non si indagherà ulteriormente sul significato di tag.

Un'ulteriore possibilità offerta della presenza di risorse condivise, poi, è il loro impiego come origine per messaggi e conversazioni: in tal senso si supponga che un documento sia una risorsa; allora potrebbe risultare piuttosto utile per gli utenti poter avviare una discussione in riferimento al contenuto dello stesso, scambiandosi opinioni e lasciando note scritte di commento. Immaginando poi che si adotti una forma di comunicazione aperta, altri utenti potrebbero quindi intervenire e contribuire a loro volta, con grande beneficio per la cooperazione. Si modifica allora la definizione di messaggio in modo tale da consentire che l'origine possa essere, appunto, una risorsa:

$$m = \langle s, p, o \rangle$$

$$u \in U, o \in M \cup R$$

8.6 Notifiche

Si introduce ora una prima vera interazione tra utenti e processi, anche se ancora in forma passiva: il meccanismo di notifiche automatiche. Si supponga ad esempio che un utente intervenga nella definizione di un processo, modificandone il flusso o anche una singola attività, in modo significativo: riprendendo l'esempio precedente per la programmazione di un meeting, un volta decisa data e orario e impostate nell'evento relativo, tutti gli utenti coinvolti dovrebbero riceverne avviso. Lo stesso si potrebbe immaginare nel caso dell'avvenuta pubblicazione di un report o ancora con l'inserimento di una deadline. **La soluzione è ancora più significativa se si ipotizza che il sistema generi notifiche anche in presenza di cambiamenti nello stato di esecuzione di un processo**.

Volendo formalizzare questa funzionalità, si definisce innanzitutto una notifica:

$$n = \langle o, u, p \rangle, u \in U$$

$$N = \{n\}$$

dove la terna specifica un'origine per la notifica, l'utente destinatario e il relativo payload. Per origine s'intende un processo, un'attività oppure anche un altro utente, ivi inclusi gruppi e classi, senza particolari restrizioni. La semantica di questa informazione è quella di permettere al destinatario di determinare la

provenienza o la causa della notifica stessa, senza necessariamente doverne interpretare il payload, o contenuto. In questa sede non se ne fornisce una descrizione più dettagliata, onde evitare di legarsi eccessivamente ad una specifica realtà o soluzione e ridurre la validità concettuale del modello proposto. Si formalizza anche l'operazione di invio di notifica:

$$notify(n, u)$$

Intuitivamente, una notifica inviata ad un insieme di utenti, sia esso un gruppo oppure una classe, sarà distribuita iterativamente ad ognuno dei suoi membri.

8.7 Feedback

Il meccanismo di notifiche automatiche permette una migliore interazione tra utenti e processi, ma si compone a tutti gli effetti come un comportamento passivo, secondo una modalità di *message pushing*. Ciò che si vorrebbe proporre, invece, è una partecipazione attiva da parte degli utenti alla definizione e all'esecuzione dei processi. In tal senso, si propone di consentire ad ogni utente coinvolto di fornire il proprio *feedback* rispetto a processi, attività e risorse. Si consideri, ancora una volta, l'esempio fin qui utilizzato della programmazione di un meeting: a causa di un'incomprensione, l'orario previsto impostato per l'evento in questione è errato. Gli utenti possono quindi lasciare il proprio feedback a margine della rispettiva attività per segnalare il problema. Similmente, prima di pubblicare ufficialmente un report si vuole ricevere conferma da parte del gruppo di lavoro circa l'effettiva completezza dell'elaborato. In tal senso, potrebbe essere più semplice ed intuitivo ricevere un feedback più strutturato e preciso, invece di commenti generici in lingua scritta. Gli esempi qui descritti rappresentano efficacemente due scenari profondamente diversi: nel primo caso si considera accettabile un feedback libero, mente nel secondo sarebbe preferibile ottenere risposte più precise e sintetiche. Si tratta di un'osservazione che verrà ripresa tra poco.

Volendo formalizzare anche questo concetto, si definisce:

$$F = \{f_i\}, f_i = \langle s, u, p \rangle$$

$$s \in P \cup T \cup R, u \in S$$

$$provide(f), f \in F$$

Similmente a quanto espresso in precedenza circa le notifiche, un feedback è assimilabile ad una terna *subject*, *user*, *payload*. Il payload è inteso esattamente come nel paragrafo precedente, senza ulteriore struttura onde evitare un legame con alcuni particolari implementazioni, mentre user è limitato ai soli utenti singoli; non è significativo ipotizzare, infatti, che possa essere un gruppo o una classe a fornire feedback, visto anche che non sarebbe poi possibili risalire al vero autore. Per quanto concerne il subject, invece, l'intenzione è di rendere ogni attività, processo o risorsa un possibile soggetto per feedback, da cui la definizione proposta.

8.7.1 Templated feedback

Riprendendo però l'osservazione precedente circa il contenuto, o payload, di un feedback, potrebbe essere maggiormente conveniente restringere la cardinalità delle risposte possibili, già predisposte. Una forma piuttosto comune è quella binaria, con il significato di positivo/negativo oppure apprezzato/non apprezzato, che si è rilevata vincente nei rapporti con gli utenti, in gran parte per la propria immediatezza d'uso e facilità di comprensione.

Formalmente, è piuttosto semplice descrivere queste casistiche, restringendo il payload ad un insieme predeterminato di possibili valori, come segue:

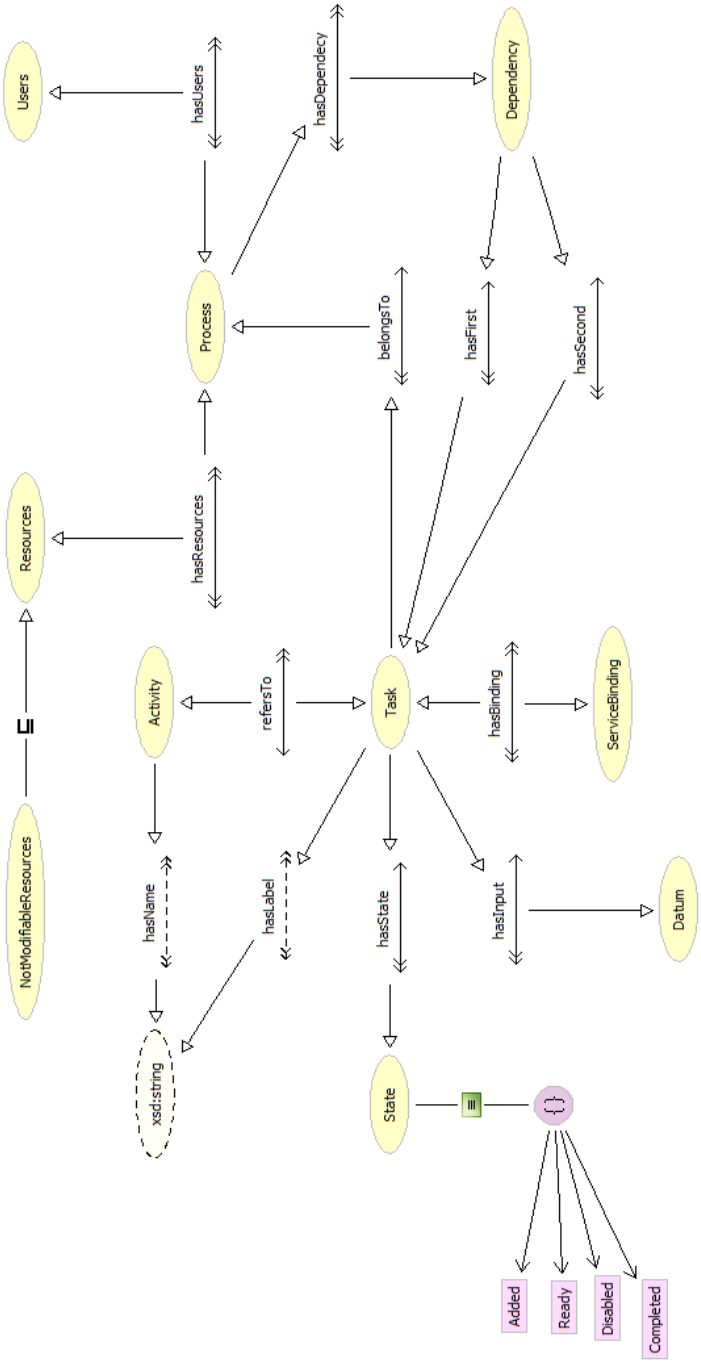
$$m =< s, p, o >$$

$$p \in P, P = \{p_1, \dots, p_n\}$$

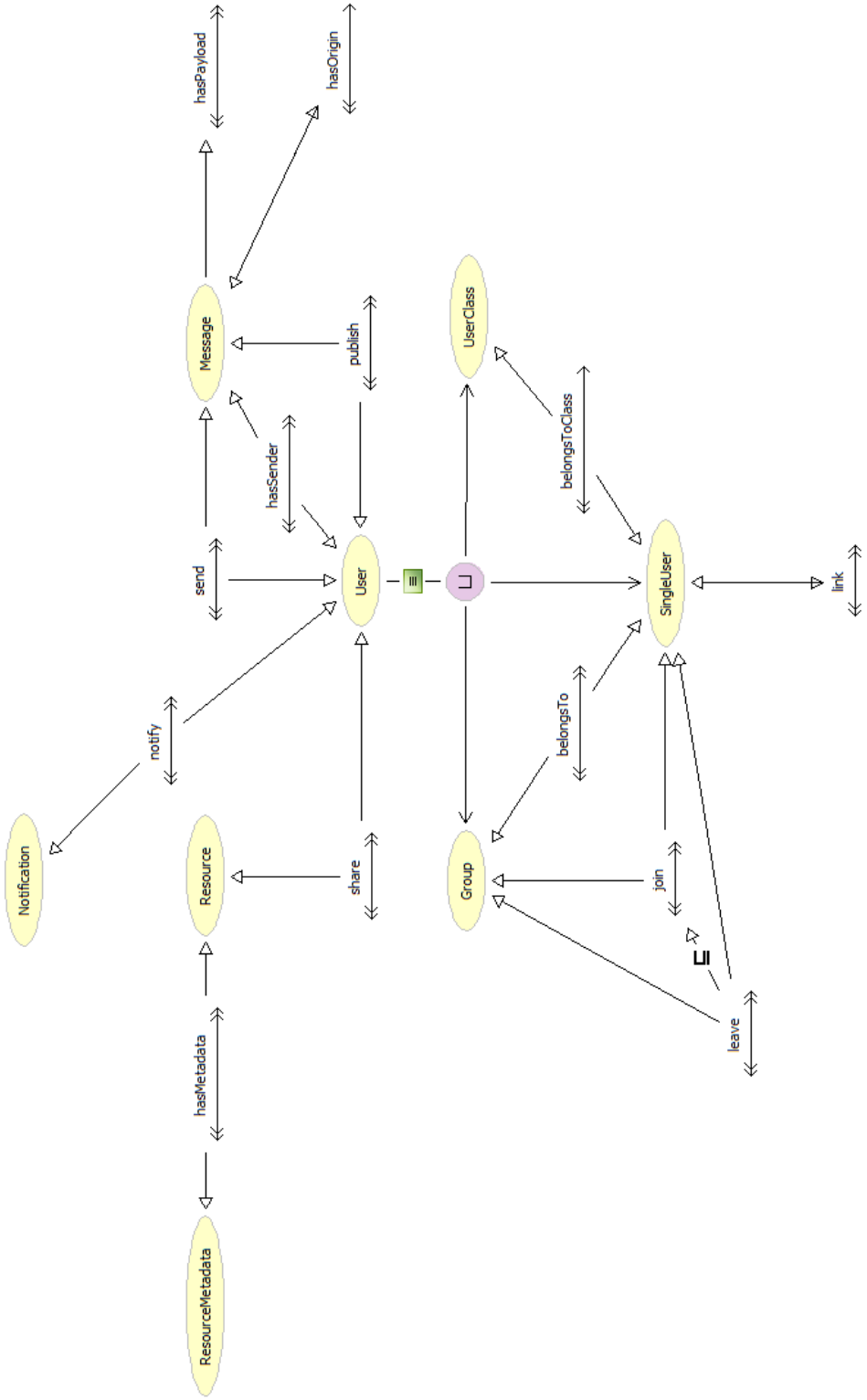
8.8 Prospetto riassuntivo

Dopo una lunga dissertazione teorica ricca di concetti e definizioni, si è ritenuto opportuno fornire un breve sintesi di quanto proposto, da intendersi anche come *reference card*, o guida rapida, quando, nel prosieguo di questo documento, verranno ripresi ed impiegati entro studi di caso realistici. In tal senso, onde evitare di introdurre una nuova simbologia o sintassi, si è deciso di esprimere i due modelli formali presentati attraverso un'ontologia, per mezzo della sintassi grafica GrOWL. Di seguito sono quindi riportati i due schemi grafici.

8.9 Processi flessibili



8.10 Processi sociali



9. Studi di caso: reti sociali attuali

9.1 Introduzione

In questa sezione saranno proposti una serie di studi di caso relativi ai più comuni e diffusi social network, con l'intento di dimostrare l'applicabilità del modello formale presentato in precedenza a prodotti concreti e largamente conosciuti. Non s'intende, tuttavia, procedere con un'analisi minuziosa di ogni aspetto di ciascun singolo social network oggetto di studio, anche in virtù della notevole velocità di cambiamento e innovazione riscontrabile in questi prodotti, cosa che renderebbe presto obsolete molte delle possibili osservazioni esprimibili.

In ultimo, si vuole sottolineare che la selezione dei prodotti è stata effettuata con lo scopo di studiare una maggiore varietà di soluzioni, e non quindi basandosi esclusivamente sulla popolarità del sito in questione. Per questa ragione, alcuni brand conosciuti e largamente utilizzati non sono stati inclusi in quanto le funzioni offerte non differiscono eccessivamente da altre soluzioni già presentate.

9.2 Facebook

Il modello adottato da Facebook rispetta la definizione di rete sociale simmetrica fornita precedentemente: gli utenti possono relazionarsi tra loro tramite un procedura di "richiesta di amicizia", che necessita peraltro di una conferma dalla controparte, al termine della quale entrambi gli utenti coinvolti possono poi accedere ai dati dell'altro. In modo analogo, un utente può unirsi a gruppi d'interesse, talvolta a fronte di un'approvazione.

Di grande interesse è invece la distinzione tra due classi, gli utenti semplici e le personalità pubbliche, aziende, istituzioni o prodotti/marchi: nel secondo caso, infatti, la partecipazione al network è diversa, in quanto la relazione con gli utenti è ottenuta in modo asimmetrico, tramite il noto meccanismo "Mi Piace". La Figura 14 ne è un esempio: si riferisce alla pagina ufficiale del Politecnico di Milano, dove è facilmente individuabile il pulsante Mi Piace, in sostituzione del più comune "Aggiungi agli amici". Le funzionalità presenti sono invece simili, includendo la bacheca annunci (sempre in figura), oltre ad una galleria di immagini e una pagina informativa.

Figura 14 Pagina ufficiale del Politecnico di Milano

Per quanto concerne la comunicazione, sono presenti sia una modalità pubbliche, tramite i noti *post* in bacheca (propria oppure di un utente "amico"), sia una modalità privata. In relazione alle risorse condivisibili, Facebook permette, ad oggi, di archiviare solamente fotografie. Sebbene sia possibile, infatti, condividere elementi per mezzo di API tramite website di terze parti, nessun contenuto è inserito concretamente nel profilo utente, limitandosi invece a pubblicare un link esterno a tale risorsa: è il caso, per esempio, di qualsiasi video.

Da notare, infine, come ogni contenuto, sia esso un post, un link o una fotografia, possa essere commentato, sia in forma di testo libero, sia tramite il meccanismo MiPiace/NonMiPiace, che rappresenta a tutti gli effetti una forma di feedback chiuso predefinito. È importante però sottolineare come il pulsante MiPiace abbia effetti diversi a seconda del contesto: ad esempio, il commento o feedback va distinto dalla relazione instaurabile con i profili pubblici descritti prima.

9.3 Twitter

Nel caso di Twitter, invece, le relazioni tra utenti sono asimmetriche: è possibile, infatti, seguire il flusso dei *tweet* di un altro utente (operazione detta *following*), ma non si tratta di un procedimento simmetrico per la controparte, che, nel caso, dovrà a propria volta "seguire" l'utente in questione. La procedura di following, poi, non richiede conferma dalla controparte. Interessante è però la

possibilità di raggruppare i vari profili seguito attraverso liste, per una migliore organizzazione e fruizione dei *tweet*. Le liste possono essere private o pubbliche.

Non esiste invece distinzione tra i profili utente, che sono tutti equivalenti da un punto di vista funzionale e d'interazione: anche aziende o istituzioni possiedono un profilo del tutto simile a quello di utenti semplici.

La messaggistica, poi, è prevalentemente a carattere pubblico: i *tweet* sono visibili anche ad utenti non registrati, mentre per comunicare in modo attivo è comunque necessario partecipare alla rete. Esiste tuttavia anche la possibilità di avviare uno scambio privato di *tweet*, all'indirizzo di un particolare utente. Di maggiore interesse è invece il meccanismo di *retweetting*, da non confondere con una più semplice risposta: per avviare una discussione, in riferimento ad un certo messaggio, gli utenti ricorrono ad pulsante *reply*, che associa quindi il messaggio ad uno scambio esistente o creandone uno nuovo; l'effetto netto è il raggruppamento di *tweet* relativi al medesimo messaggio d'origine. Al contrario, il funzionamento del *retweetting* non è inteso per partecipare ad una discussione, quanto piuttosto per inoltrare ai propri follower (gli utenti che sono iscritti al flusso del profilo in questione) un messaggio d'interesse pubblicato da un'altra fonte. Per esempio, si immagini che un utente X segue Y, ma non Z, mentre Y segue Z. Se quest'ultimo pubblica un messaggio, Y riceverà una notifica pressoché immediata, mentre Z, non essendo iscritto, no. Attraverso il retweet, allora, Y può inoltrare il contenuto del messaggio anche ai propri follower, dandogli maggiore visibilità e audience. Merita una menzione anche il concetto di preferito, applicabile ad un qualsiasi tweet, tramite cui gli utenti possono tener traccia dei messaggi di maggior interesse personale. I preferiti sono pubblicamente visibili tramite il profilo di ogni utente. La Figura 15 presenta un esempio di retweet, in cui è visibile sia l'autore sia il messaggio originale, oltre all'indicazione del retweet a lato.

Figura 15 Esempio di retweet, tratto dal profilo del Politecnico di Milano

Un aspetto decisamente particolare di Twitter è, però, la totale mancanza di hosting per risorse: anche nel caso di una semplice immagine o fotografie, infatti, è indispensabile appoggarsi ad un servizio esterno, dal quale ottenere poi

un riferimento o link da includere nei propri messaggi. Questa situazione potrebbe però mutare in futuro, in virtù di alcune recenti dichiarazioni dello staff[11].

9.4 YouTube

Per quanto possa apparire insolito, anche questo popolare sito dispone di una struttura piuttosto vicina ad un social network: innanzitutto ogni utente dispone di un proprio profilo (più propriamente un canale), e può stabilire relazioni con altri tramite le sottoscrizioni (*subscriptions*) ad altri canali. In questo modo il sottoscrivente può ricevere una notifica ogniqualvolta un nuovo elemento, un filmato cioè, viene pubblicato su un canale sottoscritto in precedenza. L'operazione non è però simmetrica. Una funzionalità poco nota, però, permette anche di condividere materiale solo con una selezione di utenti[12].

L'elemento centrale, in questo caso, è la condivisione di risorse, più precisamente materiale audiovisivo. Al di là delle specifiche funzionalità per la gestione della risorsa in quanto tale (conversione di formato, risoluzioni, etc…), l'interesse di questa trattazione si focalizza sull'aspetto sociale e in tal senso è piuttosto rilevante l'approccio disponibile per la gestione del feedback da parte degli utenti: è possibile in prima istanza scrivere un messaggio di testo libero, con la possibilità di inserire un riferimento esplicito (la "@" notation già vista in Twitter), oppure la funzionalità MiPiace/NonMiPiace, del tutto analoga a quella presente in Facebook. Questo feedback binario è però applicabile anche ai commenti di altri utenti, fatto che permette di presentare con maggior efficacia i messaggi più rilevanti o d'interesse perché maggiormente votati dagli altri utenti.

[11] http://www.ilsole24ore.com/art/tecnologie/2011-06-04/twitter-conferma-cinguettii-rete-135209.shtml?uuid=AaU5DAdD

[12] http://youtube-global.blogspot.com/2010/05/more-choice-for-users-unlisted-videos.html

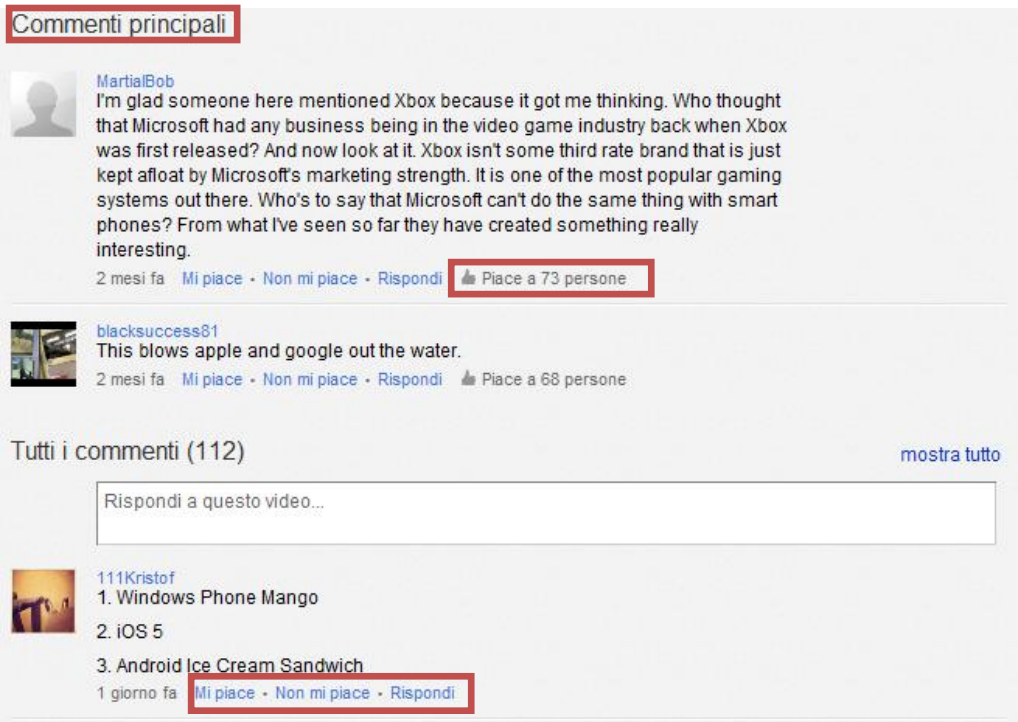


Figura 16 Esempio di video ricco di commenti. Si notano in particolare in primo piano i messaggi più votati dagli utenti.

In Figura 16 è riportato un esempio tratto da video: è evidente la sezione relativa ai commenti più votati dagli utenti, che viene proposta immediatamente prima del form per l'inserimento di un commento e della lista di tutti i messaggi. Viene anche proposta l'indicazione del numero di feedback positivi ricevuti, valore che è anche impiegato per determinare la popolarità crescente. Per ogni commento, comunque, sono sempre immediatamente disponibili i pulsanti MiPiace/NonMiPiace.

A margine di quanto detto, si segnala che è anche disponibile un semplice sistema di messaggistica privata, che consente agli utenti di comunicare tramite semplici messaggi inviati tramite il sito principale.

9.5 Google+

Trattandosi di un progetto ancora in fase di sviluppo, un'analisi approfondita di Google+ sarebbe verosimilmente destinata a divenire rapidamente obsoleta a fronte dei prevedibili cambiamenti e delle aggiunte previste nel corso dei prossimi mesi.

Allo stadio attuale (partecipazione su invito), comunque, le funzionalità principali appaiono abbastanza definite, anche se piuttosto simili a quelle presenti su altri prodotti concorrenti, come ad esempio Facebook. In tal senso, si è preferito in questa sede individuare ed analizzare piuttosto i tratti unici, a

cominciare dal concetto di cerchia (*circle* in inglese). Le cerchie rappresentano raggruppamenti di contatti definiti da ciascun utente entro il proprio profilo personale, con lo scopo di consentire una migliore e più semplice selettività nella condivisione di messaggi e contenuti. In tal senso, quindi, la distinzione tra comunicazioni pubbliche e private è più sottile e meno visibile, rappresentando in effetti casi limite nell'utilizzo delle cerchie. Ciononostante, è significativo osservare come le cerchie non siano equivalenti ai gruppi (intesi sia come concetto astratto, sia come specifiche implementazioni, quali lo stesso Facebook): le prime infatti hanno valenza all'interno di un dato profilo utente, mentre i gruppi sono oggetti indipendenti.

In merito alla distinzione tra classi d'utenza, è presente, anche se non ancora in via definitiva, un supporto specifico per aziende e istituzioni[13], inteso per evitare l'iscrizione al network di marchi o prodotti ufficialmente non consentite dai soggetti interessati.

Secondariamente, è possibile notare come la condivisione di risorse non si limiti alle sole immagini, ma includa nativamente anche filmati, sia tramite YouTube sia tramite un servizio integrato nel network.

9.6 Prospetto riassuntivo

A completamento di questa trattazione, si vuole sintetizzare in breve quanto emerso. Nella tabella seguente sono riportati i social network proposti, e per ciascuno è precisato il supporto o meno di una determinata specifica definita nel formalismo del capitolo precedente.

	Facebook	**Twitter**	**YouTube**	**Google+**
Classi	2	**1**	**1**	2
Gruppi	Sì	**No**	**No**	Sì
link	Simm.	Asimm.	Asimm.	Simm.
send	Sì	Sì	Sì	Sì
publish	Sì	Sì	Sì	Sì
share	Sì	**No**	Sì	Sì
notify	Sì	Sì	Sì	Sì
provide	Sì	Sì	Sì	Sì
Base	Relazioni	Relazioni	Risorse	Relazioni

[13] http://www.google.com/apps/intl/it/business/details.html

Come è possibile notare, esiste una forte corrispondenza tra il modello concettuale presentato e le effettive soluzioni adottate nei prodotti attualmente più noti. Alcune particolarità riguardano i sistemi che non prevedono differenti tipologie di account (classi), come Twitter o YouTube, ma questa situazione corrisponde banalmente ad un caso limite. Più interessante è invece la mancata possibilità di condividere qualsivoglia risorsa all'interno di Twitter, ma, come già, accennato questa limitazione scomparirà a breve.

Un'ulteriore osservazione riguarda infine la natura intrinseca dei vari social network: alcuni, quali Facebook o Twitter, focalizzano la propria attenzione sulle connessioni tra utenti, ivi inclusi i gruppi (o anche le cerchie, nel caso di Google+). Altri invece incentrano le proprie funzionalità sulla condivisione di risorse (tra cui anche Flickr o SlideShare, oltre allo stesso YouTube) e sul contributo e feedback da parte degli utenti circa le stesse.

10. Architettura

10.1 Introduzione

In questa sezione viene presentata un'architettura di riferimento per la definizione di una soluzione software al tema appena discusso. In Figura 17 sono presenti tutte le parti essenziali, che saranno analizzate con i dovuti dettagli nel seguito della trattazione: l'*activity catalog*, per la gestione delle attività di base, il *service registry*, per l'integrazione con servizi esterni, l'*editor* per la creazione e gestione visuale di processi, il *process storage*, per la persistenza delle informazioni, *l'engine* per l'esecuzione dei processi, il *social connector* per la fruizione di reti sociali, lo *state manager* per la gestione delle transizioni di stato della attività e un *insieme di API* di sistema per l'integrazione di software esterni.

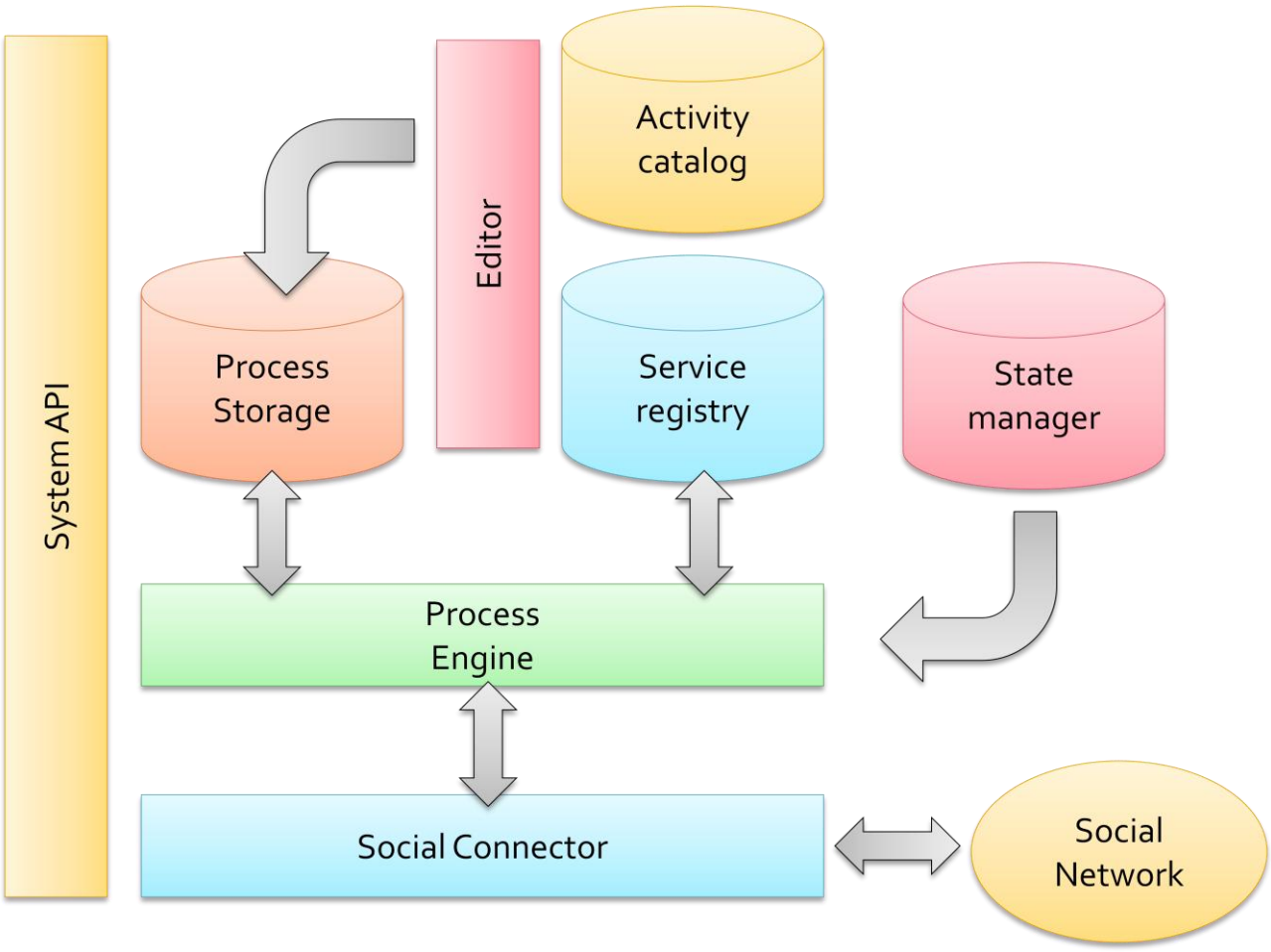


Figura 17 Architettura di riferimento

10.2 Activity catalog

Rappresenta un catalogo di *activity*, o attività elementari, che gli utenti possono utilizzare per definire i proprio processi. L'accezione qui intesa si riferisce ad un concetto maggiormente astratto, senza precisare eventuali dettagli implementativi o legati a sistemi o software reali. Per chiarire il concetto, si riportano alcune attività di esempio, che saranno peraltro riproposte anche nel seguito. Da sottolineare anche come per ciascun *activity* sia possibile impostare una serie di parametri, dipendenti dalla specifica tipologia o contesto.

Attività	Parametri di configurazione
Creazione sondaggio	Titolo, opzioni disponibili
Pubblicazione evento in calendario	Titolo, orari (inizio-fine), luogo, descrizione
Condivisione documenti	Documento, utenti coinvolti
Invio messaggio	Contenuto, destinatari

Volendo proporre come termine di paragone i noti concetti del mondo OOP, le *activity* sono assimilabili alle classi. Tuttavia, nel modello di processo qui proposto non vengono considerati altri aspetti tipici, quali ereditarietà o incapsulamento, ragion per cui occorre tener ben distinte queste due realtà non sovrapponibili.

Da un punto di vista funzionale, l'activity *catalog* è **inteso principalmente con un'entità di sola** lettura con lo scopo di fungere da base fondamentale per l'editor di creazione dei processi. Ciononostante, questa architettura non esclude la possibilità di modificare le classi di attività presenti, ad esempio aggiungendo nuove operazioni prima assenti con lo scopo di rendere l'infrastruttura più completa e funzionale. Occorre però evidenziare come l'ipotesi aggiuntiva di abilitare anche la modifica o la rimozione di activity esistenti **possa potenzialmente dare luogo a inconsistenze**, ed in particolare in riferimento a processi già definiti e persistiti nel *process storage*. Il caso forse più eclatante è probabilmente la presenza di processi per cui una o più attività inserite non sia più disponibile presso l'activity catalog. Sarebbe ovviamente possibile concepire opportune soluzioni sia nel caso si voglia prevenire il verificarsi di tali situazioni (impedendo ad esempio modifiche al catalog) sia in quello in cui si ammetta la possibilità di rendere maggiormente flessibile e personalizzabile l'elenco delle attività di base (ricorrendo alla sola cancellazione logica o al limite attraverso il versioning dei processi e del catalog), ma nessuna di queste sarà effettivamente

trattata nel seguito. Si tenterà, cioè, di focalizzare maggiormente l'accezione di flessibilità verso l'utilizzo di una molteplicità di servizi esterni, come illustrato nel paragrafo seguente.

10.3 Service registry

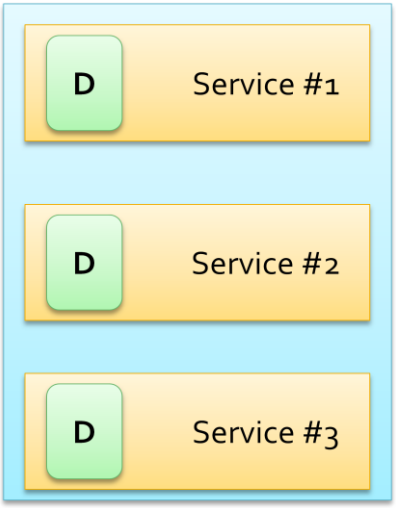
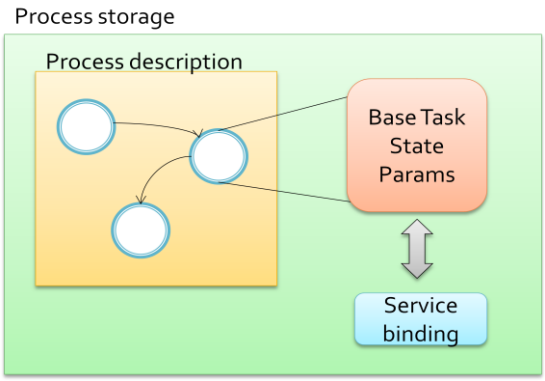


Figura 18 Struttura del service registry

Parallelamente all'activity catalog, che gestisce le tipologie di attività concettuali, il *service registry* è inteso per garantire l'integrazione con i servizi esterni e la capacità di svolgere le attività di un processo attività per mezzo delle funzionalità resa disponibili dal corrispondente provider di servizio. Più precisamente, questo componente funge in prima istanza da registro (o catalogo) dei diversi servizi supportati dal sistema, precisando in particolare quali tipologie di activity, nell'accezione prima intesa, siano disponibili. Concettualmente, infatti, la gran parte di servizi remoti oggi disponibili sono assolutamente non equivalenti, sia in termini di funzionalità erogate, sia per quanto riguarda le possibilità di configurazione e personalizzazione offerte per ciascuna di esse: ad esempio due diversi servizi possono erogare funzionalità simili, e predisporre modalità d'interazione diverse: basti pensare ad una semplice attività come la pubblicazione di un sondaggio che permetta di concordare la data e l'orario di un appuntamento; taluni servizi hanno capacità piuttosto limitate e consentirebbero solo di ottenere risposte a fronte di opzioni già definite, mentre altri lascerebbero più spazio agli utenti gestendo correttamente le diverse disponibilità orarie di ciascuno.

Il service registry mantiene pertanto una descrizione concettuale di ogni servizio riconosciuto, in una struttura dati denominata *service descriptor*: oltre alle già citate corrispondenze tra funzionalità e activity, però, è inserito anche un rifermento al componente software interno adibito all'integrazione concreta con il servizio remoto. Tale aspetto è particolarmente critico in relazione all'effettiva esecuzione di processi, cui corrisponde in ultima analisi un'invocazione remota al fornitore del servizio, punto che tipicamente richiede un supporto tecnologico specifico e diverso da scenario a scenario. La gestione di tali problematiche viene quindi delegata al livello applicativo sottostante e non è quindi oggetto di questa iniziale trattazione teorica.

10.4 Process storage



L'ultima componente funzionalmente associata alla persistenza delle informazioni è il *process storage*, la cui competenza riguarda la gestione dei processi composti dagli utenti e il relativo stato una volta avviati. Conseguentemente, il primo aspetto si relaziona maggiormente con il componente *editor*, garantendo le funzioni necessarie a mantenere una descrizione completa e aggiornata di ogni processo, con riferimento alle *activity* richiesti, alle informazioni parametriche di configurazione di ciascuna e alle scelte di associazione con servizi registrati nel sistema attraverso il service registry precedentemente illustrato. Quest'ultimo punto si rende necessario in quanto il sistema deve consentire la massima flessibilità possibile agli utenti in fase di progettazione, in particolare la possibilità di fruttare appieno quanto offerto dai servizi esterni, senza però miscelare indistintamente parametri generali di una particolare classe di activity e caratteristiche specifiche di un dato provider.

La persistenza dello stato di esecuzione di un processo, e conseguentemente lo stato proprio di ogni attività in esso inclusa, è invece da ricondurre al *process engine*, per il quale è di fondamentale importanza poter determinare non solo la struttura particolare dei processi da avviare, intesa sia in termini di attività sia di configurazione, ma anche lo stato attuale di ogni singolo *task*, secondo la specifica definita dal modello formale. A ciò occorre poi affiancare la necessaria configurazione per l'invocazione del servizio remoto, da tradursi in una precisa chiamata effettuata proprio dall'*engine* sottostante. Viste queste considerazioni, è anche lecito parlare di *self-descriptive processes* [26], ovvero processi la cui serializzazione definisca completamente un processo, sia a livello statico (attività e loro proprietà, flusso di esecuzione, etc…), sia a livello dinamico circa lo stato di esecuzione delle singole attività.

10.5 State manager

Si tratta di un componente di fondamentale importanza, in grado di gestire in modo unificato la gestione dello stato di esecuzione di un processo e delle singole attività in particolare. Come osservato in precedenza durante la definizione del meta modello di processo, è necessario mantenere lo stato di un'attività per ciascuno dei propri attori. In aggiunta, nel caso di dipendenze da

attività assegnate a più utenti, è necessario anche fornire un meccanismo per la risoluzione di tali dipendenze, che tenga traccia di quanti attori abbiano svolto l'attività di oggetto.

Lo state manager opera in tal senso, mantenendo una struttura dati aggiornata e gestendo l'accesso concorrente da parte degli altri componenti di sistema.

10.6 Editor

Giunti a questo punto, è possibile introdurre una prima parte degli aspetti funzionali della soluzione, e più precisamente la definizione di processi. L'*editor* è il componente essenziale concepito per questo scopo nel rispetto del modello formale precedentemente illustrato: lo strumento consente creare un processo componendo tra loro attività di base tra quelle rese disponibili dall' activity catalog, secondo i costrutti di connessione previsti dal sistema. Similmente, anche la gestione di un processo esistente, e precisamente la riconfigurazione di una specifica attività, è resa possibili per mezzo di una user interface adatta: l'utente può inizialmente scegliere su quale processo intervenire, accedendo così al process storage tramite un apposito componente dedicato, lo *storage bridge*. La selezione delle activity disponibili, così come l'associazione e la configurazione ad un servizio esterno specifico è invece prerogativa dell'*activity browser*, in grado di agire come interfaccia al activity catalog, e del *service binder*, cui competono invece i passi necessari ad associare un servizio esterno ad una data activity inserita nel processo corrente.

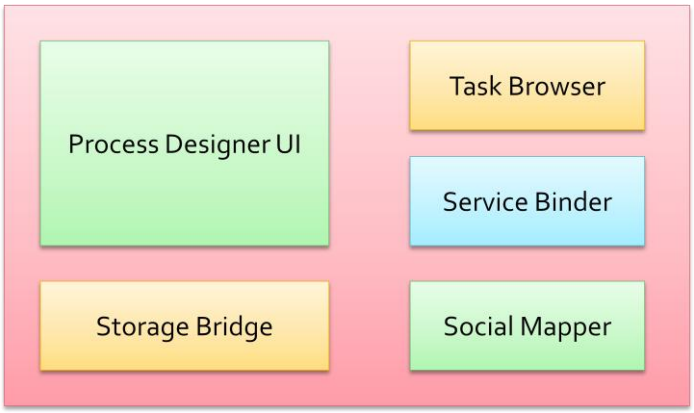


Figura 19 Sottocomponenti principali dell'editor di processo

In ultimo, la gestione della connessione ad un social network, necessaria per supportare i processi sociali, è affidata al *social mapper*, che consente la selezione e la configurazione del servizio desiderato dall'utente.

10.7 Process engine

In ultima istanza si trova il *process engine*, la seconda componente proattiva del sistema e l'unica in grado di operare in modo autonomo, seppur entro i limiti imposti dalla definizione del modello formale. Come ampiamente illustrato in precedenza, lo svolgimento, o esecuzione, di un processo definito e "salvato" all'interno dell'area di *process storage* si fonda sulla partecipazione degli utenti, i quali possono genericamente fornire feedback, in diverse forme, o interagire direttamente con ciascuna attività secondo modalità prefissate, in particolare in accordo con la configurazione fornita e la classe specifica di attività considerata: con riferimento al precedente esempio circa la pubblicazione di un sondaggio per la definizione degli orari di un ipotetico meeting, tale attività sarà tipicamente ristretta ai soli utenti interessati o coinvolti e prevederà, come comportamento prefissato, la votazione della (o delle) opzioni preferenziali da parte di ciascun partecipante.

Da una prospettiva maggiormente funzionale, il process engine accede al process storage per ottenere il descrittore del processo richiesto, quindi determina lo stato di ogni attività coinvolta e abilita contestualmente le operazioni di interazione possibili, quali l'aggiunta di attività, la modifica di esistenti o l'avvio e l'arresto di task già predisposti, secondo le modalità indicate nel modello teorico. In quest'ultimo scenario, specialmente, l'engine procede interagendo con il service registry per richiedere una chiamata a servizio remoto, precisando ogni parametro, generico del modello o specifico del servizio, necessario. In questo stadio subentra quindi la componente software attiva del service registry adibita ad comunicare con il servizio richiesto, la quale tipicamente esegue una chiamata remota via web. Quando l'operazione termina il risultato, o più correttamente la risposta del servizio invocato, viene restituita ed interpretata affinché sia comprensibile da parte del process engine, scongiurando la necessità di configurare quest'ultimo per analizzare correttamente il formato di risposta di ciascun servizio. Infine, in base al risultato ottenuto (usualmente successo o errore, talvolta un'informazione specifica dipendente dal servizio e dal tipo di operazione svolta), l'engine completa il flusso qui descritto comunicando con lo *state manager* per notificare il completamento dell'attività per l'utente corrente.

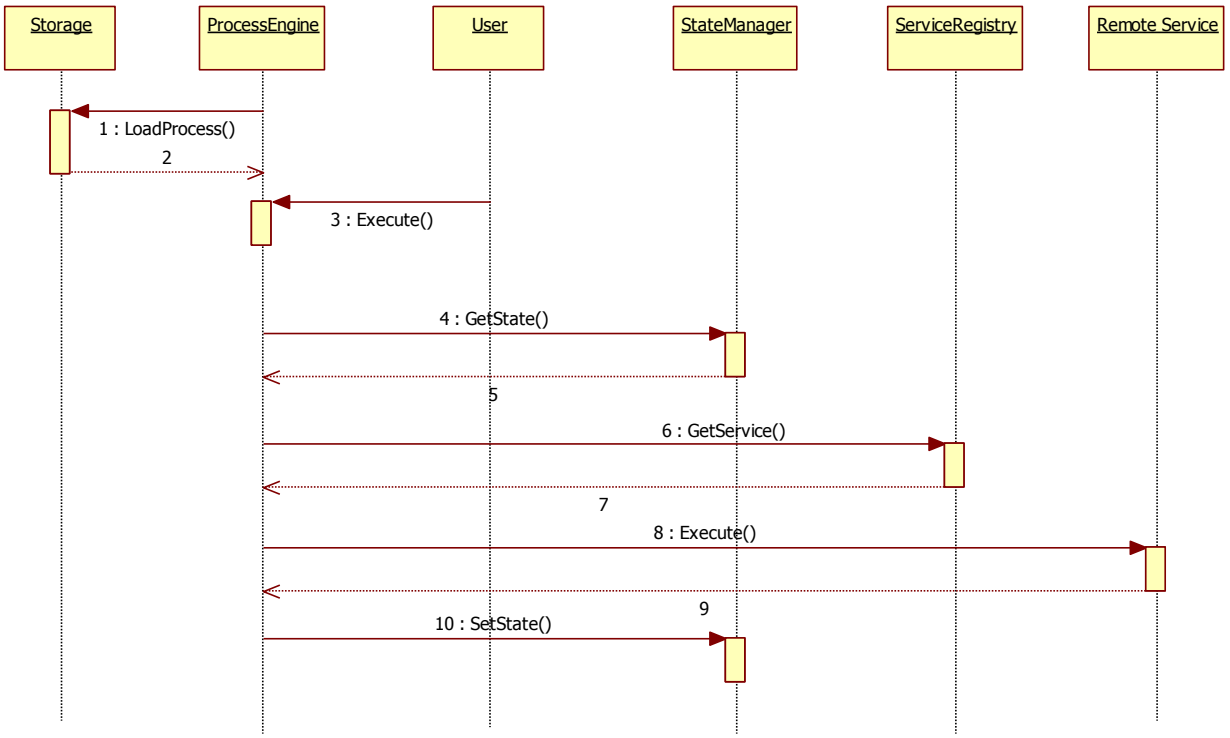


Figura 20 Flusso di esecuzione di un'attività da parte del process engine

10.8 System API

Definito il funzionamento del componente principale, il process engine, è d'interesse introdurre la possibilità di usufruire delle relative funzionalità per mezzo di una opportuna API. La situazione più ricorrente si presenta nel momento in cui s'intenda sviluppare uno o più client per i processi in esecuzione nel sistema, in modo da garantire agli utenti una user interface più omogenea ed usabile, possibilmente fruibile anche attraverso dispositivi mobili. È verosimile ipotizzare anche che siano costruire anche UI specifiche per un determinato processo, onde fornire la migliore interattività possibile per ciascuno specifico scenario.

10.9 Social Connector

Il supporto ai processi sociali è invece prerogativa di un ulteriore sezione appositamente concepita per lo scopo, il *social connector*. In riferimento al modello discusso nei capitoli precedenti circa le funzionalità generali presenti in un social network, il social connector rende disponibili tali operazioni in modo astratto senza necessariamente possedere una conoscenza dell'effettivo social network utilizzato dal processo, garantendo quindi un forte disaccoppiamento

strutturale e grande flessibilità nella gestione dei propri processi a base sociale. La tabella qui presente ne fornisce un breve riassunto:

Funzione	Commento
Link	Connessione tra due utenti
Join/Leave	Adesione/abbandono di un gruppo
Send	Invio di un messaggio privato
Publish	Invio di un messaggio pubblico
Share	Condivisione di una risorsa
Notify	Invio di una notifica
Provide	Invio di un feedback

Si prospetta tuttavia una problematica già analizzata in fase di definizione dei processi flessibili, circa la necessità di assolvere agli usuali parametri di configurazione specifici per accedere ad ogni particolare istanza di social network considerata. Nondimeno, esiste anche il requisito particolare di integrare concretamente l'interfacciamento con un applicazione web esterna tramite le usuali API a disposizione. Per tali ragioni, la struttura interna del social connector può figurarsi nel modo seguente:

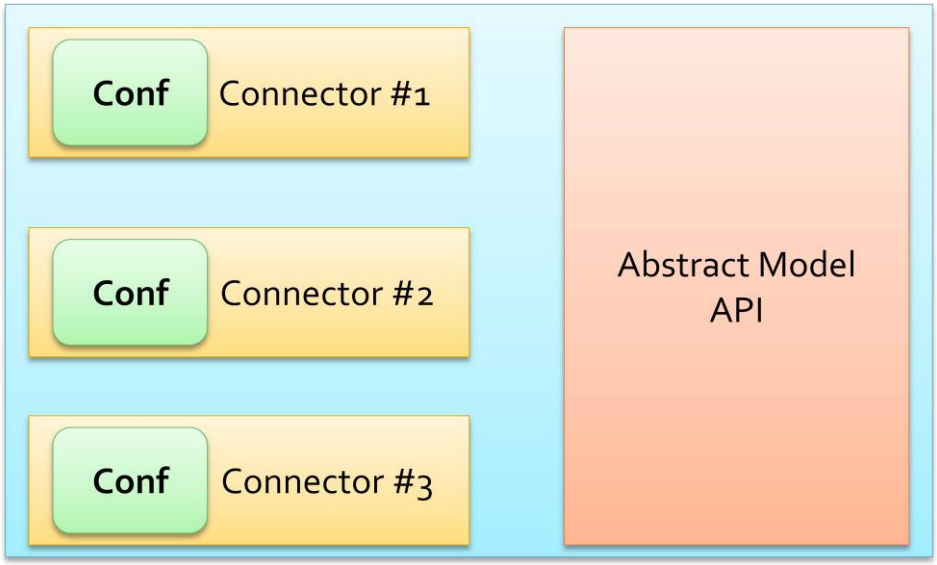


Figura 21 Strutta interna del social connector

Il blocco denominato *abstract model API* corrisponde in prima approssimazione alle operazioni definite nel modello formale e rappresenta a tutti gli effetti l'interfaccia esterna primaria del componente in questione. I

blocchi etichettati come *connector*, invece, identificano il supporto per l'integrazione e l'accesso a diversi social network esterni. È altresì importante sottolineare come per ciascuno di essi sia prevista una configurazione specifica, necessaria ad assolvere requisiti di applicazione specifici, come tipicamente l'autenticazione utente. È significativo notare, tuttavia, come non vi sia imposto alcun vincolo circa l'associazione con un solo, singolo servizio esterno, e che pertanto è del tutto ammissibile predisporre il sistema all'utilizzo di più social network in contemporanea, anche nell'ambito di uno stesso processo utente assegnato.

Parte III
Dominio specifico
Alta Scuola Politecnica

11. Studio di caso

11.1 Introduzione

In questa seconda parte della trattazione si vuole sperimentare l'applicabilità ad un contesto operativo specifico dei modelli precedentemente discussi, onde misurarne il reale grado di adattabilità e generalità teorizzato nella relativa definizione. La scelta è ricaduta sul programma di studi dell'Alta Scuola Politecnica, da un lato per il coinvolgimento personale di chi scrive, ma dall'altro anche e soprattutto per le potenzialità offerte come scenario reale per la definizione e modellazioni di processi rispetto a requisiti concreti e consolidati.

Inizialmente, quindi, saranno presentati due scenari tipici, nella forma di *user stories*, per fornire un'introduzione sintetica al contesto di studio ASP (Alta Scuola Politecnica), per poi passare successivamente ad una loro analisi funzionale e giungere in ultimo alla formalizzazione dei due processi attraverso gli strumenti teorici presentati nei capitoli precedenti. In particolare sarà anche proposta una serie di specifiche per l'implementazione reale di un sistema software adatto, in cui saranno inserite anche alcuni mockup per chiarificare la soluzione ideata.

11.2 Processo A: gestione *school*

Una school è una sessione di seminari, della durata di cinque giorni, cui tutti gli studenti ASP debbono partecipare. Essendo organizzate in località scelte di volta in volta, è necessario che gli studenti effettuino una registrazione in cui confermino la loro presenza ad ogni seminario o evento in programma, oltre a specificare se intendono usufruire dei servizi di logistica predisposti o meno. La registrazione viene aperta dalla ASP Board non appena il programma della settimana è stato deciso dai docenti incaricati di organizzare le attività. Gli studenti devono allora ricevere una notifica di apertura della sessione e completare la procedura entro la data di scadenza prevista. Inoltre, sebbene non si tratti di un'attività ufficiale, gli studenti devono normalmente fornire una distinta per la composizione delle camere nella struttura ospitante.

Successivamente, durante lo svolgimento della school, i docenti condividono con gli studenti una serie di documenti inerenti le tematiche discusse, tipicamente da utilizzare al termine dei seminari per lo svolgimento

dei lavori di gruppo assegnati di volta in volta. In tal senso, i docenti, e gli eventuali tutor, suddividono gli studenti in gruppi inviando poi loro comunicazione, anche in merito ai punti di ritrovo stabiliti per gli incontri. Al termine ogni gruppo deve produrre alcuni risultati, tipicamente in formato elettronico, come documenti o presentazioni, da condividere con i colleghi e i docenti.

Parallelamente, esiste poi la necessità di fornire comunicazioni organizzative, ad esempio il cambio di orario per i pasti o le indispensabili indicazioni per il checkout durante l'ultimo giorno di permanenza. Al termine della sessione, poi, gli studenti sono tenuti a compilare un questionario di feedback.

11.3 Processo B: progetto multidisciplinare

Nel corso dei due anni della durata del programma, gli studenti devono svolgere un progetto specifico, scelto alla presentazione dei corsi, in un gruppo composto dalla ASP Board secondo le preferenze espresse da ciascuno studente (questa fase non è tuttavia d'interesse in questo scenario). Lo svolgimento del progetto dipende in gran parte dal contesto specifico, ma esistono tuttavia una serie di attività piuttosto comuni, che coinvolgono sia gli studenti del gruppo sia i tutori accademici e aziendali che seguono il progetto: innanzitutto, è piuttosto frequente che entrambe le parti debbano condividere documenti e risorse, tipicamente materiale di riferimento o elaborati prodotti. Di norma gli studenti, o un sotto-gruppo di questi, ricevono un assegnamento da uno o più docenti, quindi lavorano attivamente alle richieste ricevute, comunicando con il docente in caso di necessità. Infine, una volta completato il compito, condividono l'elaborato finale con tutti i membri del gruppo di lavoro per le correzioni e/o la discussione dei risultati ottenuti. Spesso, poi, esiste la necessità di organizzare meeting in presenza o via Internet, verificando le disponibilità di ciascuno, condividere il materiale da presentare o analizzare e definire gli obiettivi da conseguire successivamente.

In particolari occasioni, poi, è necessario presentare alcuni documenti ufficiali presso la ASP Board: si tratta di report circa lo stato di avanzamento del progetto, che devono essere compilati entro una scadenza precisa.

11.4 Analisi e modellazione

Si procede ora con un'analisi formale e dettagliata degli scenari appena descritti, utilizzando gli strumenti teorici esposti in precedenza. Per ciascuno scenario saranno quindi specificati: il modello di processo, con una breve descrizione per le attività più significative, gli utenti e le risorse coinvolte. In particolare, però, sarà dedicato diverso spazio alla definizione dell'insieme delle attività disponibili in un ipotetico activity catalog.

12.4.1 Processo A

Nel primo caso, tre classi di utenti sono coinvolte: i docenti responsabili della organizzazione della school, la board ASP e gli studenti. Si desidera impiegare classi invece che gruppi dal momento che i ruoli rivestiti sono chiaramente distinti e non sovrapponibili.

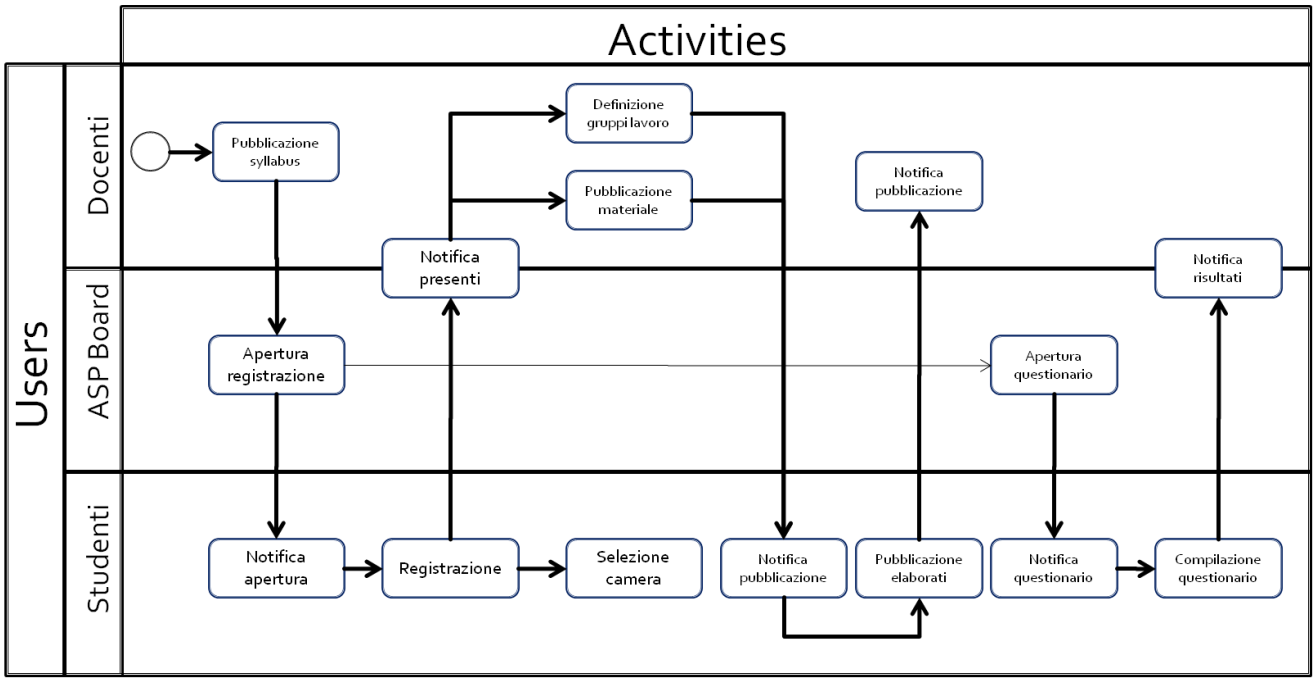


Figura 22 Schema di processo dello scenario A

Il processo è quindi schematizzato in Figura 22, dove le attività sono separate in base alle classi di utenza coinvolte (si noti che le attività poste sulle linee di separazione s'intendono condivise e dirette ad entrambe le classi). Alcuni commenti circa le attività principale e alcune scelte effettuate:

- La pubblicazione del programma della school è un requisito per l'apertura della registrazione dal momento che gli studenti sono tenuti ad indicare la propria presenza a margine di ogni seminario o evento previsto.

- Sono indicate esplicitamente le notifiche da inviare ad ogni classe o gruppo di utenti, proprio per indicare anche schematicamente i destinatari corrispondenti.

- È bene ricordare come un'attività debba soddisfare prima le proprie dipendenze prima di poter essere avviata, ma in ogni caso sarà l'utente corrispondente a procedere con l'attivazione. Per questo motivo, l'apertura del questionario di valutazione del corso ha sì una sola dipendenza, ma certamente non sarà completata se non previa conferma dell'ASP board al termine della school.

- Esistono diversi percorsi paralleli, con il significato intuitivo di poter soddisfare contemporaneamente le dipendenze di più attività ed abilitarle simultaneamente. Pertanto, una volta notificati gli studenti presenti, i docenti e i tutor posso contemporaneamente procedere con la pubblicazione del materiale e la preparazione dei gruppi di lavoro.

Per quanto concerne le risorse coinvolte, queste possono essere identificate principalmente come documenti di testo, PDF o presentazioni visuali, più raramente come materiale multimediale. In particolare, poi, alcuni di questi sono certamente da considerarsi di sola lettura, come ad esempio il materiale di riferimento fornito dai docenti.

12.4.2 Activity catalog

La formulazione proposta, naturalmente, non è fissata e immutabile, quanto piuttosto una prima realizzazione dello use case descritto precedentemente. Per questo motivo, è decisamente più significativo fornire una collezione di attività base, unitamente ad alcuni service binding, per consentire agli utenti di personalizzare se necessario l'intero processo così come specifiche attività, secondo gli specifici requisiti o condizioni che potrebbero emergere in futuro o in casi particolari.

Attività	Servizio	Funzionalità	Note
Condivisione file	SlideShare	Condivisione privata	Richiede utenza Pro
	DropBox	-	Utenza base limitata
Selezione camere	Google Docs	SpreadSheet	
Registrazione,	SurveryMonkey	-	

Questionario	Murvey	-	Versione beta
Comunicazioni	Twitter	Tweeting	Solo messaggi brevi
	Facebook	Post in bacheca	

12.4.3 Social network

La scelta della rete sociale cui appoggiarsi non semplice, poiché deve tener conto, in primo luogo, del numero di utenti raggiungibili tramite esso: se solo una parte di questi possono venire coinvolti, un solo servizio non è probabilmente sufficiente. Secondariamente, un altro fattore da tenere in stretta considerazione è la possibilità di riutilizzare il prodotto in questione come service binding per altre attività nel task catalog, semplificando anche le problematiche d'integrazione a livello applicativo.

Per queste ragioni, la scelta per questo scenario è ricaduta su Facebook, che non solo offre la più grande base di utenti al mondo, ma è anche uno strumento piuttosto conosciuto e familiare a molti, e dispone far l'altro di innumerevoli applicazioni in grado di accedervi ed integrar visi. Le funzionalità previste sono quindi le stesse descritte in precedenza, che sono riportate qui per semplicità:

Operazione	Funzionalità	Note
Join/leave	Gruppi	I gruppi possono essere chiusi
Send	Messaggi privati	
Publish	Post in bacheca	
Notify	Notifiche	Automatiche/built-in
Feedback	Commenti/MiPiace	

In alternativa, anche Google+ sarebbe stato una scelta interessante, ma allo stato attuale di sviluppo sono ancora molte le funzionalità da definire e perfezionare, senza considerare poi che la base di utenti oggi presenti è ancora piuttosto limitata.

12.4.4 Processo B

Come nel caso appena discusso, anche qui viene riportato per primo uno schema di processo di riferimento, lasciando sempre però la possibilità che gli utenti intervengano per apportare modifiche o correzioni secondo necessità.

In questa situazione, le classi di utenti coinvolte sono gli studenti membri di un dato gruppo di progetto, i tutori accademici di riferimento e partner

industriali supporto. Ancora una volta, si tratta di ruoli distinti e non mutabili, ragion per cui non si applica il concetto di gruppo; il quale, però, è decisamente adatto a gestire le suddivisioni operative decise dai tutor: tipicamente, infatti, ogni progetto prevede al suo interno gruppi di minor estensione o anche squadre distinte con obiettivi differenti (specie nei progetti più numerosi). In questi casi, sussiste la necessità di una, seppur minima, gestione gerarchica, che può essere risolta con l'uso dei gruppi.

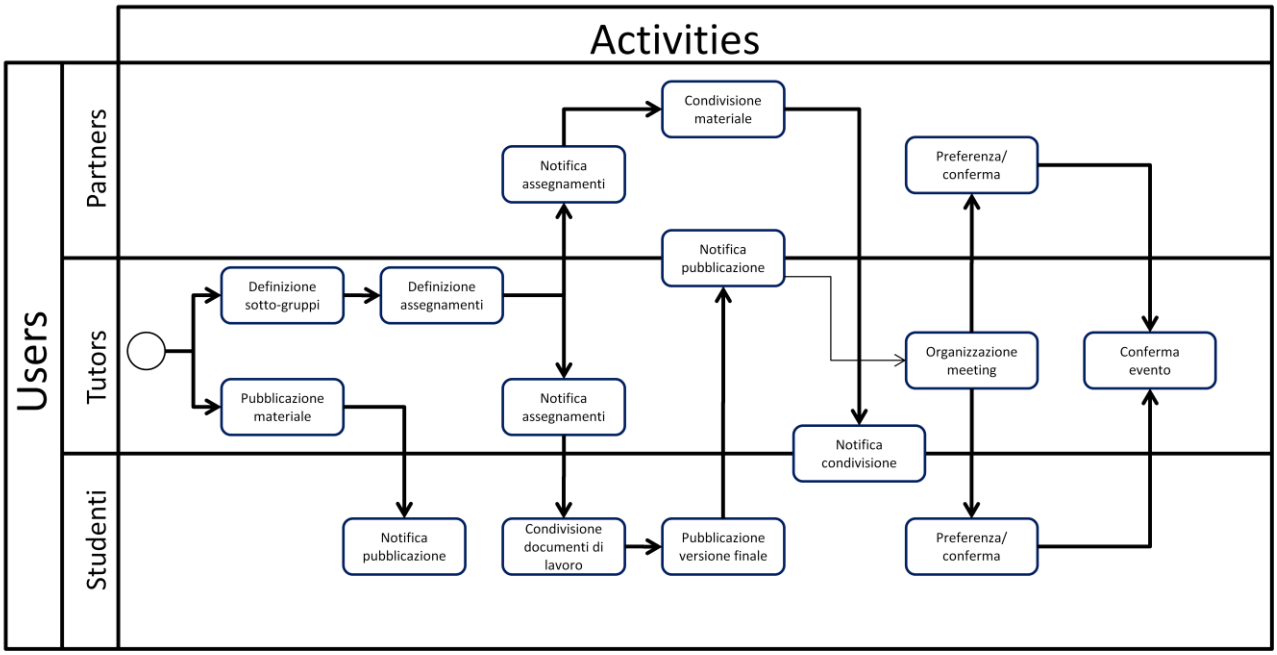


Figura 23 Schema del processo B

In relazione all'identificazione della risorse coinvolte, invece, oltre ai documenti già citati nel processo A, in questo caso è possibile aggiungere anche entità più astratte, come ad esempio meeting o eventi: si è già accennato, infatti, come sia interessante considerare alcune entità come risorse di processo onde consentire agli utenti di fornire il proprio feedback in merito; fatto, questo, che si estende naturalmente anche ai documenti condivisi e che in precedenza ha rivestito un interesse solo marginale. Nell'ambito del lavoro di progetto ha invece grandissima importanza potersi confrontare su tematiche ben precise, organizzando le discussioni e i conseguenti scambi di messaggi per argomento. In tal senso, anche la funzione di messaggistica riveste un'importanza riveduta, dal momento che la durata e gli obiettivi del progetto implicano una notevole mole di compiti da svolgere, con conseguente produzione di materiale e informazioni che il sistema deve poter gestire e proporre agli utenti in modo congeniale.

12.4.5 Activity Catalog

Attività	Servizio	Funzionalità	Note
Condivisione file	SlideShare	Condivisione privata	Richiede utenza Pro
	DropBox	-	Utenza base limitata
Gestione eventi	Doodle	-	Solo pianificazione
	WhenIsGood	-	Solo pianificazione
	Google Calendar	-	

12.4.6 Social network

Vista la particolare natura dei progetti, e volendo gestire un certa soglia di riservatezza circa i compiti assegnati, ricorrere ad una soluzione più attenta alla confidenzialità dei dati e dei documenti potrebbe rivestire un ruolo cruciale, specie nel grado di adozione ed utilizzo da parte degli utenti più accorti.

Per queste ragioni si considera l'utilizzo di Yammer, scelta favorita anche dal fatto che il prodotto in questione opera utilizzando indirizzi email con lo stesso dominio per definire la rete sociale privata.

12. Realizzazione

12.1 Introduzione

In questo capitolo viene presentata la realizzazione di una soluzione software concepita secondo l'architettura presentata in precedenza. Inizialmente saranno discusse alcune importanti scelte di *software engineering* circa l'architettura del sistema, tra cui le risoluzione della reciproche dipendenze tra i blocchi architetturali e la gestione del relativo *lifecycle*. In ultimo saranno proposti alcune schermate rappresentanti il prototipo durante il funzionamento con alcuni processi e attività dimostrative.

12.2 Requisiti specifici

La realizzazione del sistema è stata studiata attentamente per essere adattata alla soluzione architetturale proposta in precedenza, in modo da trarne il massimo vantaggio.

Innanzitutto, si evidenzia immediatamente una fitta interdipendenza tra i vari componenti presenti mostrati in Figura 17: si tratta di un sistema fortemente modulare in cui ogni blocco funzionale ricopre un ruolo altamente specifico e necessita di cooperare con altri blocchi per assolvere i propri compiti. È quindi necessario stabilire un metodo per risolvere tali dipendenze e rendere pienamente operativo ciascuno blocco funzionale.

Secondariamente, mentre l'aspetto funzionale di ogni blocco è stato precisato da un punto di vista concettuale, l'effettiva implementazione potrebbe non essere univoca: si pensi ad esempio al process storage, che potrebbe limitarsi ad utilizzare semplici file XML oppure ricorrere ad un DBMS. Lo stesso dicasi per lo state manager. Poiché esistono profonde differenze tra ogni approccio, e in generale non esiste necessariamente una scelta superiore alle altre, sarebbe opportuno rendere l'intero sistema assolutamente componibile, in modo da permettere la naturale sostituzione di un blocco con uno funzionalmente equivalente, senza però richiedere particolari modifiche.

Allo stesso modo, infine, esiste la criticità di integrare moduli esterni, in special modo *wrapper* per l'invocazione di servizi o l'accesso ai social network. Non è infatti ipotizzabile né di limitare i servizi disponibili solamente a quelli già

sviluppati, né tantomeno di essere costretti a sostanziali modifiche del codice di sistema per aggiornamenti o variazioni delle API remote.

12.3 Principi e *design pattern*

I problemi elencati possono essere efficacemente risolti ricorrendo ad un noto principio di ingegneria del software, il *Dependency inversion principle*[14].

Si specificano così le dipendenze di ogni componente tramite un contratto software, tipicamente una interfaccia (nel senso OOP) e si delega un "attore" esterno (un componente software di libreria) affinché risolva autonomamente tali dipendenze *a run-time* (non staticamente, quindi). Questo meccanismo, o più propriamente *design pattern*, è noto come *dependency injection*[15] ed è disponibile in numerose librerie. Grazie ad esso è quindi possibile risolvere il primo requisito specifico.

Inoltre, le soluzioni di dependency injection sono spesso gestite sinergicamente con un Inversion Of Control (IoC) Container: queste tecniche, che includono diversi altri design patter (factory, service locator, etc…) permettono di gestire un particolare oggetto, il contenitore; per prima cosa, i componenti (classi più in generale) sono registrati nel contenitore, che annota i contratti (interfacce) e le dipendenze (comunemente analizzandone il costruttore). Successivamente, è possibile interrogare il contenitore stesso affinché restituisca un'istanza adatta per un dato contratto, assicurandosi di soddisfare ogni dipendenza.

L'aspetto interessante di questo approccio è la possibilità di ricorrere alle comuni tecniche di caricamento dinamico di codice eseguibile per mezzo della *reflection*. In tal modo, è possibile accedere a package o assembly esterni (non inclusi nel sistema base) e caricarne gli oggetti in essi definiti a run-time, realizzando di fatti una soluzione *a plugin*. È infatti sufficiente definire una porzione di codice, identificabile, adibita alla registrazione dei tipi nel container di cui sopra per estendere a piacere gli elementi a disposizione del sistema, come ad esempio implementazioni alternative dei blocchi architetturali o diversi client per servizi esterni.

Da segnalare, infine, come l'impiego di questi accorgimenti contribuisca significativamente a semplificare il testing dell'applicazione, in particolare lo

[14] http://www.objectmentor.com/publications/dip.pdf
[15] http://www.martinfowler.com/articles/injection.html#FormsOfDependencyInjection

unit testing dei singoli componenti, garantendo così una migliore robustezza finale.

12.4 Sviluppo

Lo sviluppo della soluzione vera e propria, quindi, è stato svolto applicando le considerazioni espresse sinora creando un gruppo di progetti .NET, tramite il linguaggio C# 4.0. Un primo progetto contiene il sistema vero e proprio, con l'implementazione di ogni blocco funzionale. Una serie di progetti secondari, invece, contiene i wrapper (uno per progetto) per diversi servizi e API esterne, ivi inclusi i connettori per alcuni social network. Infine, l'interfaccia web, unitamente ad una API RESTful di backend, è stata realizzata tramite ASP.NET e il pattern MVC (Model-View-Controller).

Di seguito sono mostrate e commentate alcune catture da schermo del risultato finale. In Figura 24 è proposta la pagina principale per la gestione dell'esecuzione di un processo. L'utente attualmente autenticato ha selezionato un processo in cui compare come attore, e può così visualizzare le attività di sua competenza. Per ciascuna di esse è riportato il titolo, lo stato e alcune informazioni di base, tra cui il servizio associato per l'esecuzione e le eventuali dipendenze. È logicamente possibile avviare solamente le attività le cui dipendenze sono soddisfatte, per il cui pulsante *Run* è visibile nella prima attività soltanto. D'interesse è poi il tasto *Feedback* che permette di commentare l'attività corrispondente, leggendo anche i commenti degli altri attori (Figura 26). Nell'esempio riportato è stato usato Twitter come social network per lo scambio di messaggi privati (*direct messages*). Si noti in particolar modo l'uso degli *hashtag* per identificare l'oggetto del messaggio, nel caso un'attività del processo.

A margine delle attività, sono presenti due sezioni con informazioni aggiuntive sul processo: tutti gli attori coinvolti e le risorse integrate nel processo. Facendo click sul nome di un attore è possibile vederne i dettagli ed eventualmente scambiarvi dei messaggi (Figura 25).

In Figura 27 è riportata invece la pagina di consultazione del service registry, che mostra tutti i servizi disponibili, oltre ai social network disponibili per l'integrazione, tutti comunque caricati come plugin esterni. È anche possibile installare e configurare nuovi servizi.

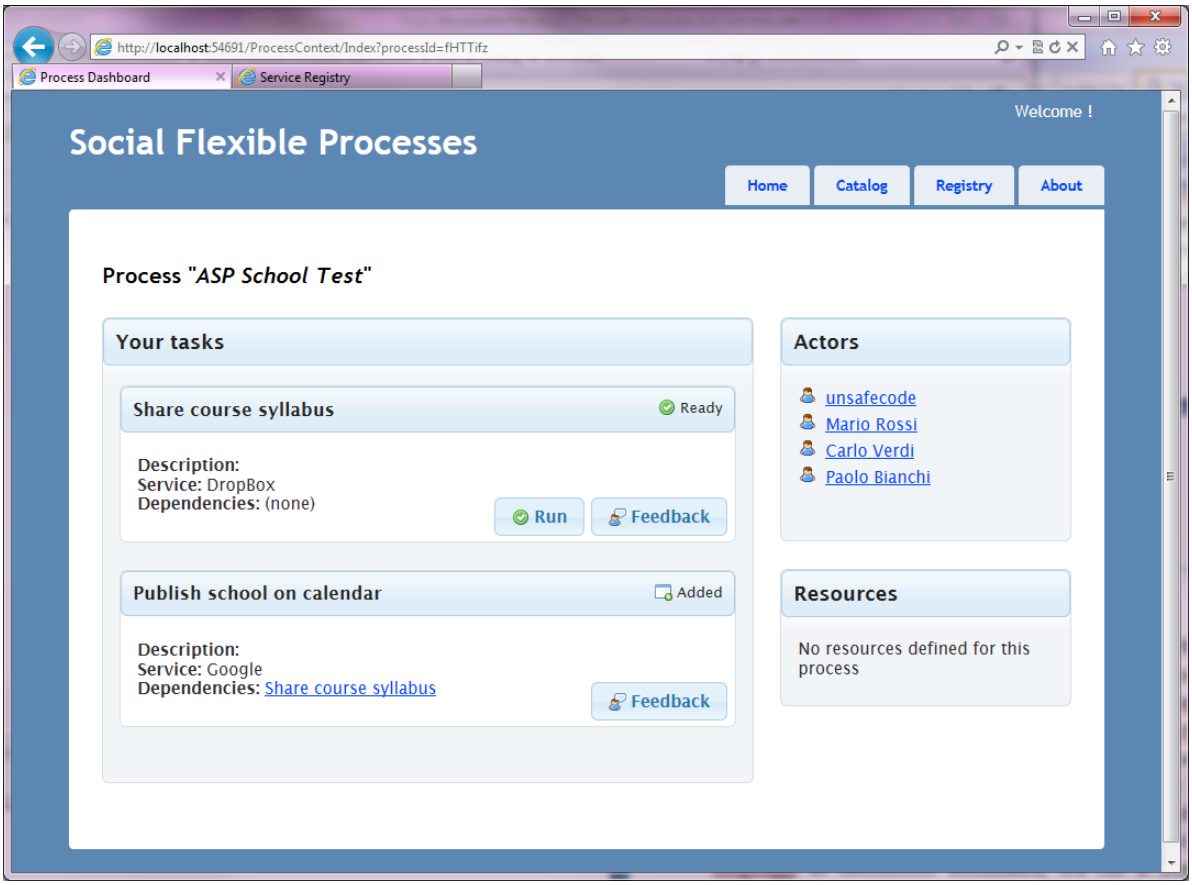


Figura 24 Gestione di processo per un attore

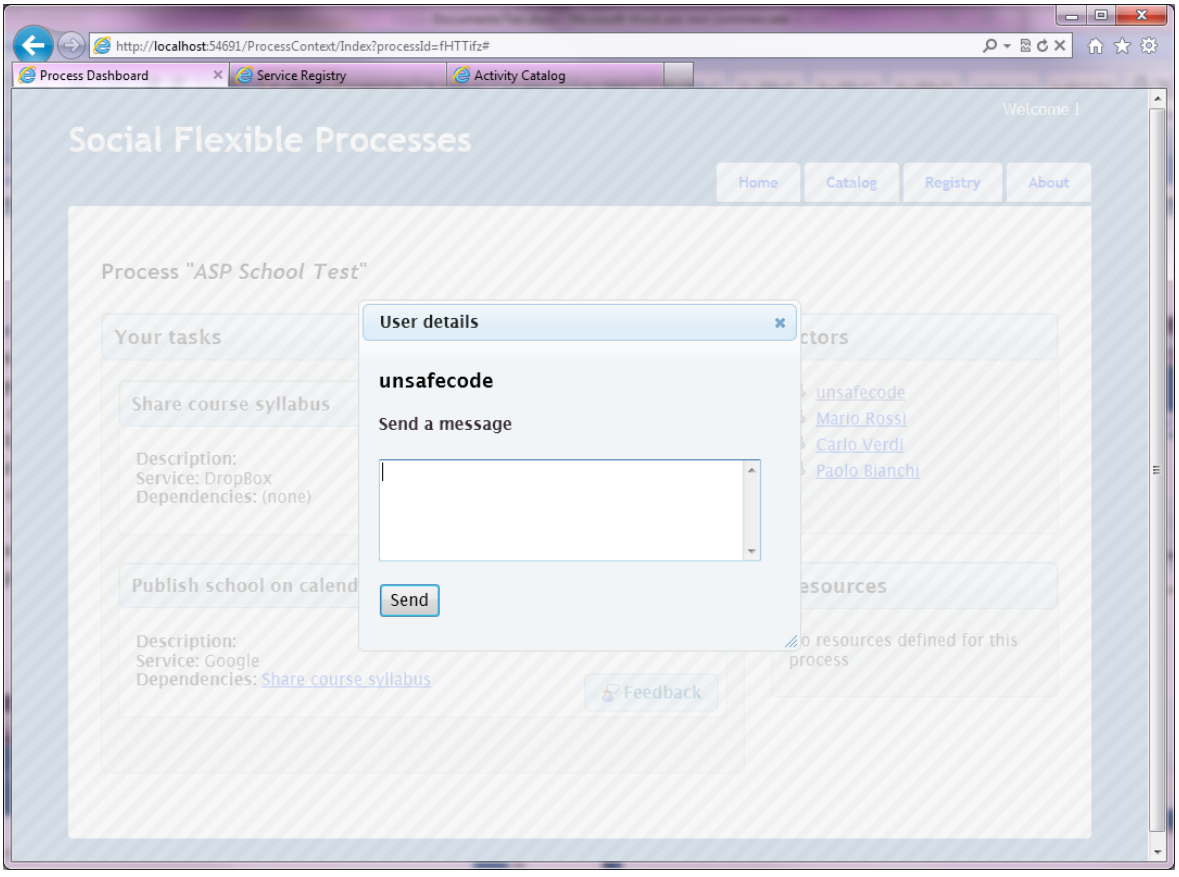


Figura 25 Scambio di messaggi tra attori di processo

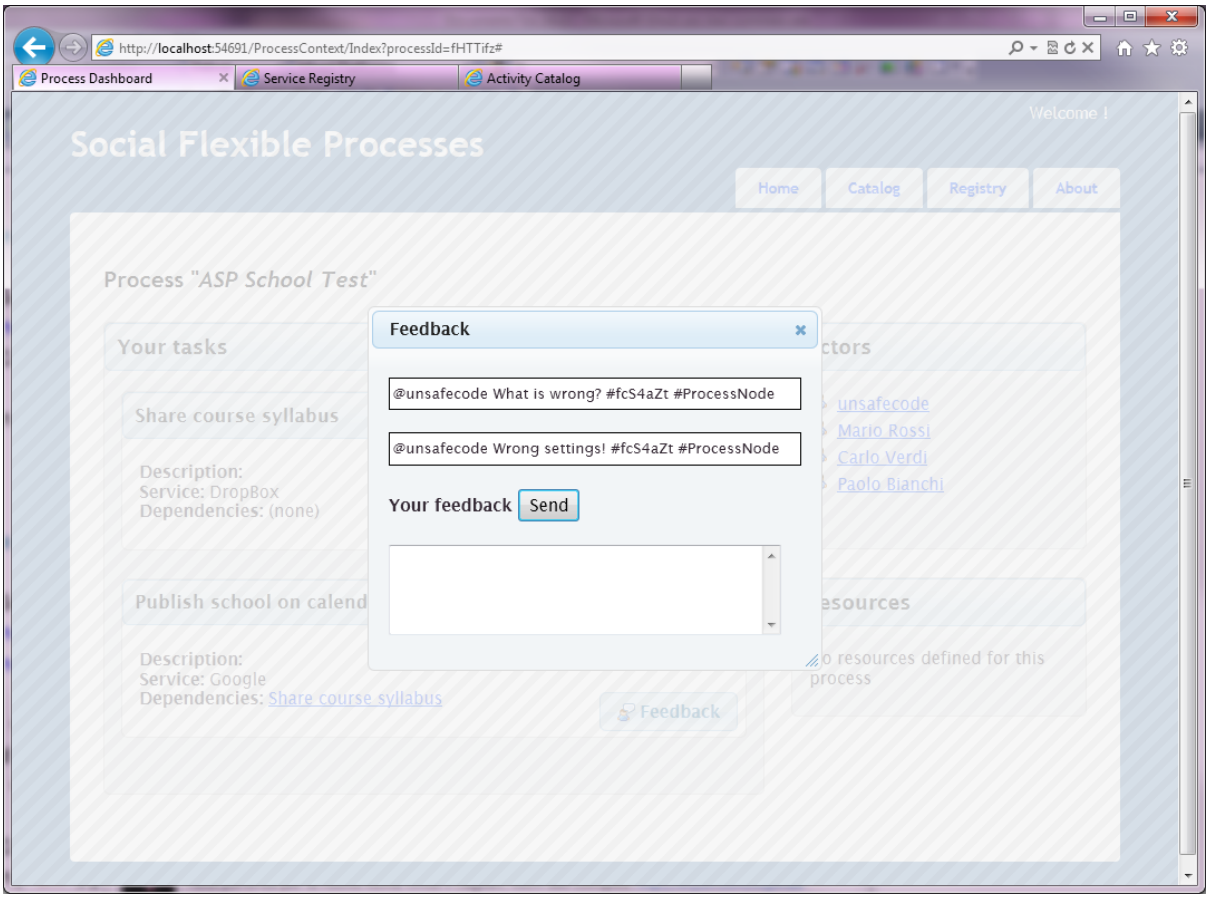


Figura 26 Feedback e discussione per un'attività utilizzando Twitter come social network

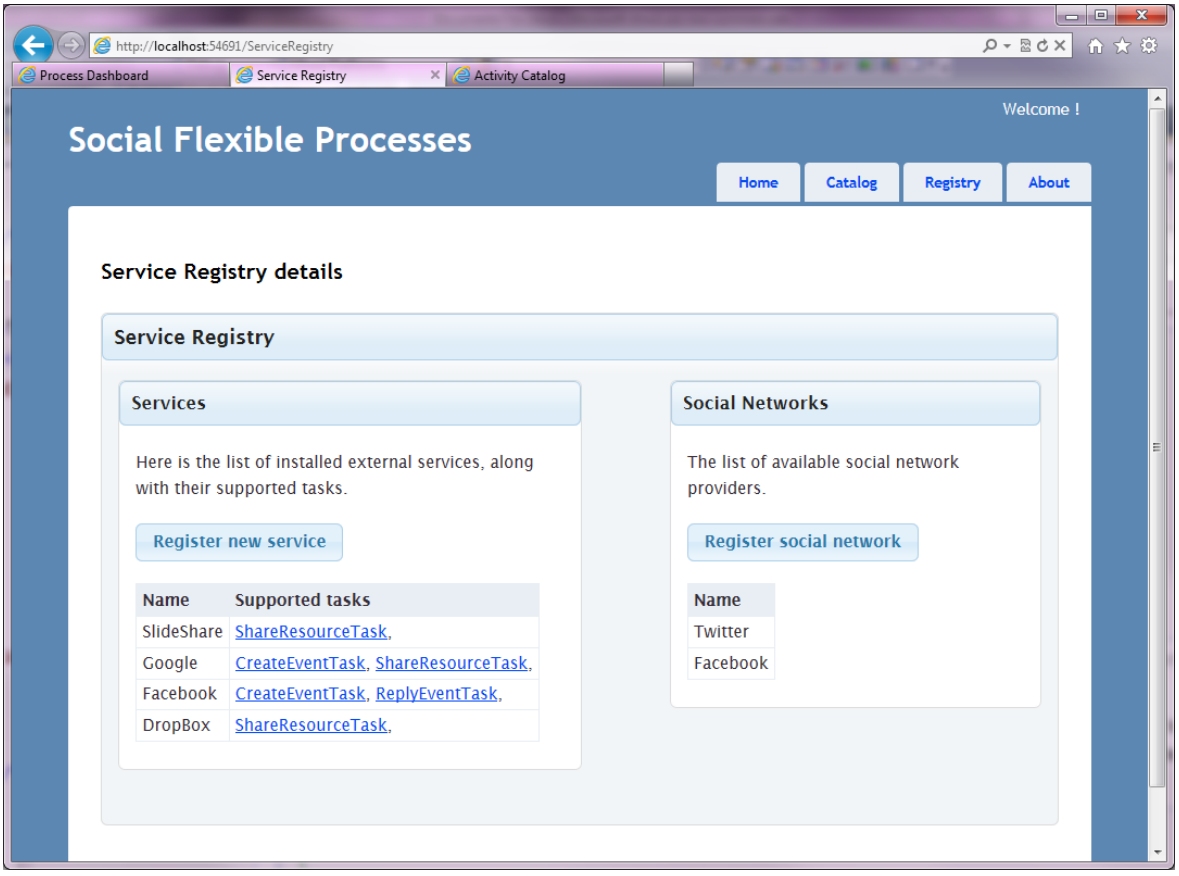


Figura 27 Pagina di esplorazione del service registry

Vale la pena effettuare alcune osservazioni circa la pagina amministrativa del *service registry*, proposta nella schermata precedente. Come accennato nell'introduzione, all'avvio del sistema sono caricate dinamicamente tutte le librerie esterne presenti nella cartella di configurazione. Per ciascuna di queste, viene invocato un oggetto adibito al caricamento e alla registrazione dei servizi e dei tipi (classi) presenti nella libreria. A questo punto, il registro può interrogare l'IoC container per determinare tutte le implementazioni di servizio (che estendono un data interfaccia) per elencarle all'utente, ma non solo: trattandosi di codice vero e proprio, è anche possibile chiedere direttamente all'istanza del wrapper di ritornare le attività supportate. Questa tecnica elimina anche la necessità di disporre di descrittori su file per i servizi.

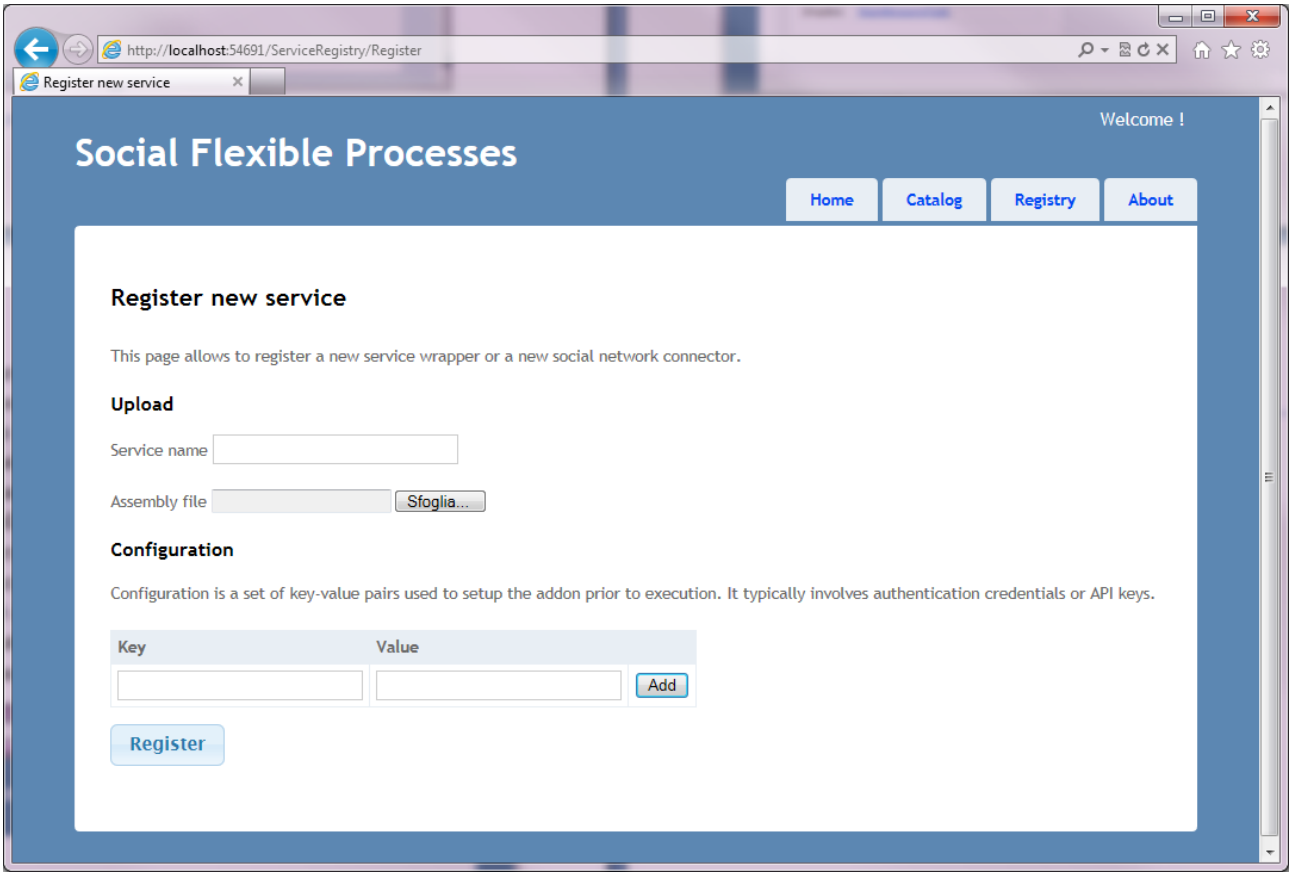


Figura 28 Registrazione di un nuovo servizio

In Figura 28 viene invece riportato il form per la registrazione di un nuovo servizio. Oltre ad un nome e alle indispensabili chiavi di configurazione (tipicamente credenziali di autenticazione o API key), è necessario specificare un file: si tratta dell'*assembly* .NET che contiene il wrapper da installare. Vale la pena sottolineare che tale *plugins* deve essere sviluppato tenendo conto delle specifiche del sistema centrale, o più precisamente implementando le interfacce base, necessarie al riconoscimento. Non trascurabile, poi, è la richiesta di un

modulo di caricamento, ovvero una classe adibita ad identificare tutti i tipi (classi) da caricare nel sistema centrale. Questo blocco può essere realizzato come un IoC Module[16].

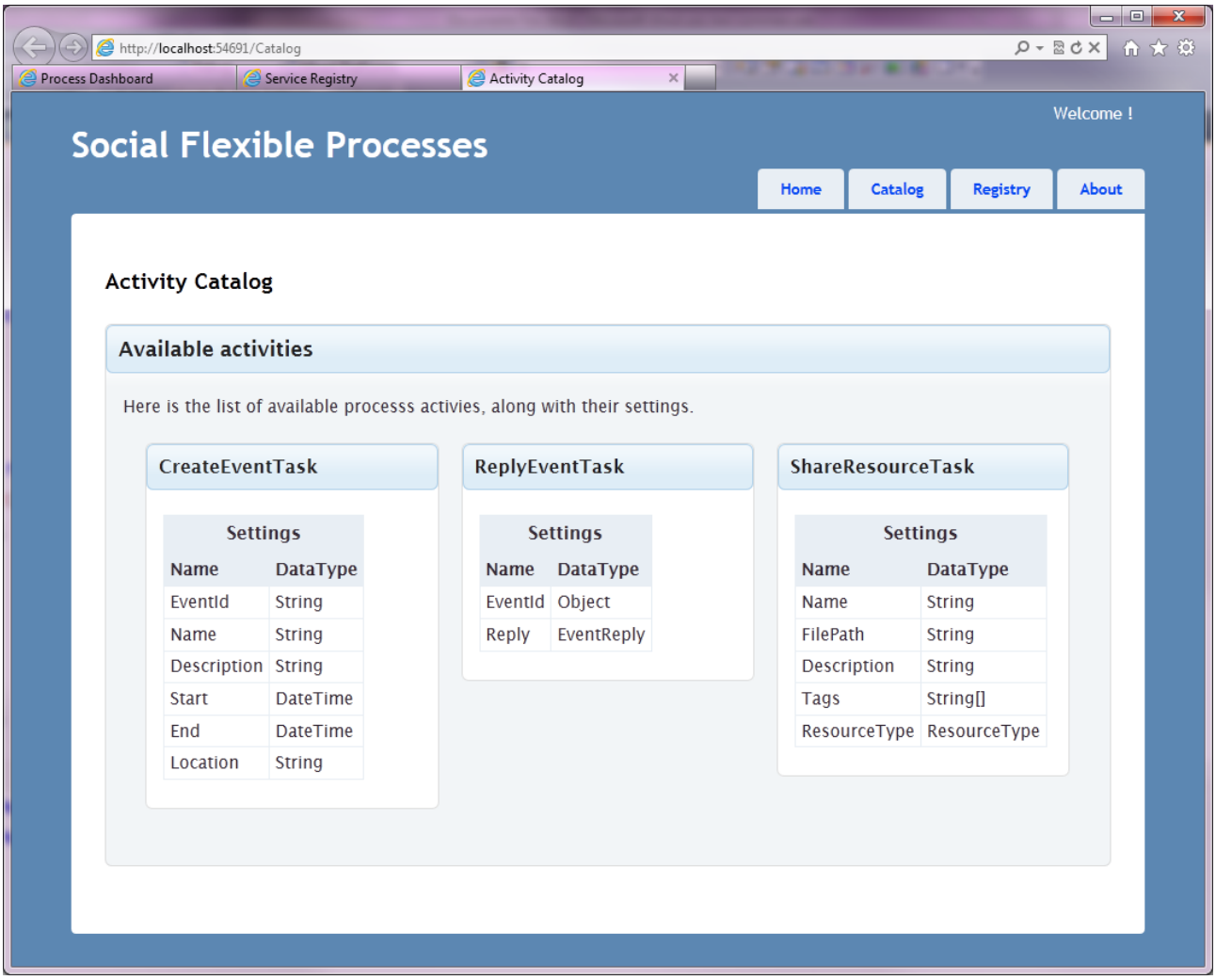


Figura 29 Consultazione dell'activity catalog

Infine, in Figura 29, è possibile osservare la consultazione dell'activity catalog. Anche in questo caso viene fatto ampio uso della *reflection*[17]: tutte le attività estendono una classe base, affinché sia identificabili correttamente dal sistema. Successivamente, però, possono inserire un qualsiasi numero di proprietà di configurazione, denominate e propriamente tipizzate (si notino gli array e le date). Similmente al caso dei servizi, anche nuove attività possono essere registrate nel sistema, includendole nella fase di registrazione svolta nei moduli IoC.

[16] http://code.google.com/p/autofac/wiki/StructuringWithModules
[17] http://en.wikipedia.org/wiki/Reflection_(computer_programming)

13. Conclusioni

In quest'ultimo capitolo si vuole riassumere brevemente quanto ottenuto durante lo svolgimento di questa tesi, e proporre quindi una serie di possibili ulteriori sviluppi, sia limitandosi a completare la soluzione presente, sia esplorando tematiche completamente nuove.

13.1 Risultati

A seguito dell'obiettivo iniziale di sviluppare le due dimensioni di "flessibilità" e "socialità" nell'ambito di processi collaborativi, le soluzione proposte, architetturale ed implementativa, rispondono ai requisiti iniziali: è stato infatti possibile definire processi composti da attività la cui gestione non solo sia affidata ad un servizio esterno, ma anche predisporre più alternative per l'esecuzione di ogni task. Sul fronte della dimensione sociale, il sistema permette innanzitutto agli attori di processo di scambiarsi agevolmente comunicazioni e di condividere risorse; ma soprattutto rende possibile gestire il *feedback* di questi ultimi in modo strutturato, esplicitando chiaramente gli "oggetti" (processi, attività, risorse, etc.) coinvolti.

13.2 Estensioni

Ciononostante, il lavoro può ancora essere esteso: per prima cosa sarebbe necessario integrare un maggior numero di servizi e definire nuove attività, così da facilitare notevolmente la creazione di nuovi processi. Le difficoltà, in tal senso, sono insite nella necessità di sviluppare wrapper *ad hoc*, determinata dalla scomparsa o dallo scarso uso, ormai, di descrittori di servizio, come il Web Service Description Language (WSDL). In tali condizioni, infatti, non è possibile procedere ad una integrazione automatica tramite generatori automatici, senza dimenticare poi gli onnipresenti requisiti specifici di sicurezza: la grande maggioranza dei servizi, infatti, richiede particolari protocolli di cifratura e/o di autenticazione (come OAuth [62]). Può invece dirsi limitato l'impatto per il riutilizzo delle interfacce standard del sistema o l'implementazione della registrazione di servizi. Lo sviluppo di una repository estesa di servizi richiede anche un'analisi approfondita dello specifico dominio di applicazione. Aldilà della messa a punto di tecniche per l'integrazione di nuovi servizi nella piattaforma, rimane quindi la necessità di individuare le attività tipiche, quindi i servizi a supporto di tale attività. Anche a fronte di una repository di servizi ben

popolata, rimane inesplorato il problema di individuare pattern di interazione tipici, in modo da poter potenziare categorie di servizi ritenuti più utili, in quanto più utilizzati, dagli utenti finali della piattaforma. Questo è possibile applicando tecniche di mining per l'individuazione di "pattern sociali" frequenti dai processi già definiti ed eseguiti.

Un diverso aspetto da considerare è l'editor di processi, il cui completamento richiederebbe un adeguato studio di usabilità e la progettazione di una u*ser interface* di facile utilizzo, in modo da garantire anche ad utenti meno esperti all'accesso alle funzionalità offerte dal sistema. Seguendo questo medesimo ragionamento, poi, sarebbe opportuno svolgere una serie di test di utilizzo con un campione composta da due tipologie di utenti: i *process designer*, con i compiti di definire il processo e monitorarne lo stato di completamento, e gli utenti finali, per i quali è di fondamentale importanza che la fruizione delle funzionalità e la partecipazione ai processi sia semplice ed efficace, prestando particolare attenzione all'importanza dello scambio di contributi e *feedback*. In particolare, sarebbe di grande interesse uno studio condotto sulle capacità del sistema di adattarsi anche a scenari diversi, ad esempio in ambito aziendale tramite servizi interni.

Ultimo, ma non per questo meno importante, sarebbe interessante esplorare in che modo risorse (strutturate) di processo possano essere create a partire dal feedback (non strutturato) di attori che collaborano tramite paradigmi sempre più diffusi quali il crowdsourcing.

Appendice

14. BPM Notation

14.1 Introduzione

Business Process Model and Notation (BPMN) è uno standard per la modellazione di processi industriali, e fornisce una notazione grafica per definire processi aziendali in un Business Process Diagram, basandosi su tecniche di *flowcharting* molto simili ai più noti Activity Diagram del mondo UML. L'obiettivo di BPMN è facilitare la gestione dei processi aziendali, proponendo una notazione che sia al contempo semplice e intuitiva ma comunque in grado di rappresentare realtà complesse. La specifica BPMN fornisce inoltre una corrispondenza tra la sintassi grafica e i costrutti dei languaggi di esecuzione sottostanti, tra cui il Business Process Execution Language in particolare.

In tal senso, quindi, BPMN è concepito per l'applicazione ai soli processi aziendali, trascurando il supporto necessario ad altre realtà, quali la gestione di strutture organizzative, la scomposizione funzionale o i modelli dei dati.

14.2 Elementi principali

La modellazione attraverso BPMN è guidata da semplici diagrammi composti da un numero limitato di elementi grafici. Ne esistono quattro categorie:

Categoria	Elementi
Flow Objects	Events, Activities, Gateways
Connecting Objects	Sequence Flow, Message Flow, Association
Swimlanes	Pool, Lane
Artifacts	Data Object, Group, Annotation

Nel seguito ognuno di questi elementi sarà analizzato.

14.3 Flow objects

I flow objects sono i principali strumenti di descrizione e prevedono tre elementi essenziali:

Figura 30 Le diverse notazioni grafiche per gli eventi BPMN

- **Evento:** un evento è identificato da un cerchio e denota il verificarsi di una data situazione (a differenza delle attività che definiscono un'azione ormai ultimata). Le icone all'interno del disco specificano la tipologia di evento, ad esempio una busta per i messaggi e un orologio per il tempo. Un ulteriore classificazione prevede eventi di cattura (nel senso che potrebbero necessitare di catturare un messaggio *per avviare* il processo) o di lancio (inteso come la possibilità di inviare un messaggio *al termine* del processo). Esistono poi dei simboli speciali: **l'evento iniziale**, denotato con un bordo sottile, agisce come punto d'ingresso del processo. Logicamente può essere soltanto un evento di cattura. Complementarmente esiste anche un **evento finale**, che identifica il risultato del processo; è rappresentato con un bordo spesso ed è classificato come evento di lancio. Ogni altro evento è detto **intermedio** ed è presentato con un bordo doppio sottile. Possono essere sia di cattura sia di lancio.

Figura 31 Simboli per task e sub-process in BPMN

- **Attività:** un'attività è indicata con un rettangolo dai bordi arrotondati e identifica un compito da svolgere. Un **task**, in particolare, rappresenta un compito atomico e indivisibile, nel senso che un ulteriore livello di dettaglio necessiterebbe di tracciarne i passi, il che esula dagli obietti di BPMN. Più complesso è invece un **sub-process**, che rappresenta a tutti gli effetti un processo a sé stante, con anche eventi di inizio e fine separati. Non è consentito, tuttavia, che il processo padre comunichi direttamente con il figlio, ad esempio collegandosi ad una sua attività interna. In ultimo, esiste anche il tradizionale concetto di transazione, che si focalizza sul concetto di atomicità. Graficamente, si distinguono dai **sub-process** per il bordo doppio sottile.

Figura 32 Esempi di gateway in BPMN

- **Gateway:** lo scopo dei gateway, disegnati come rombi, è quello di determinare la scissione o l'unione di percorsi d'esecuzione sulla base di determinate condizioni, in analogia a costrutti simili tradizionalmente presenti nei diagrammi di flusso.

14.4 Connecting objects

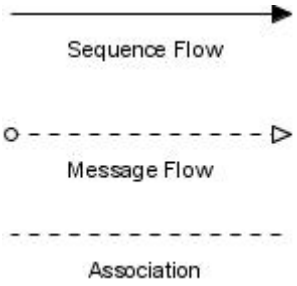


Figura 33 I tre connecting objects definiti in BPMN

Le connessioni tra i diversi flow objects sono rappresentate per mezzo dei *connecting objects*, di cui esistono tre elementi:

- **Sequence flow:** disegnato come una freccia a tratto continuo, mostra l'ordine in cui eseguire le attività. Il simbolismo è poi arricchito da alcune aggiunte: un rombo all'inizio della freccia indica uno tra i possibili flussi condizionali a partire da un'attività, mentre la presenza di un barra (*slash*) definisce il flusso predefinito (sempre nel caso di alternative condizionali).

- **Message flow:** i flussi di messaggi sono disegnati invece con una freccia tratteggiata, al cui inizio è posto un cerchio vuoto e con una punta vuoto al termine. Lo scopo di questo elemento è di segnalare quali messaggi attraversino i limiti organizzativi, come ad esempio le *pool*. È bene evidenziare, comunque, che un flusso di messaggio non può mai essere usato per collegare attività o eventi nella stessa pool.

- **Associazione:** le associazioni vengono impiegate per correlare artefatti o testo semplice a flow objects. Opzionalmente, è possibile indicarne un

verso tramite un freccia a punta vuota: ad esempio, è possibile identificare un artefatto come input o output a seconda del verso del collegamento. Questa opzione non è ammessa però per sequence o message flow, dato che questi elementi presentano già un'indicazione di direzione e verso.

14.5 Swimlanes

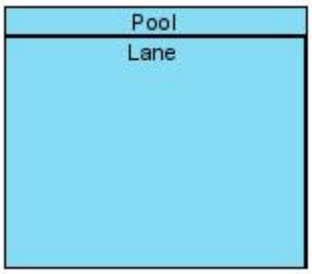


Figura 34 Pool e lane

Le *swimlanes* sono una tecnica visuale per organizzare e raggruppare le attività sulla base degli attori coinvolti. In BPMN sono presenti due elementi in questa categoria:

- **Pool:** una pool identifica uno tra gli attori principali di un processo, tipicamente seguendo la struttura dell'organigramma aziendale. Una pool contiene poi al suo interno diverse *lane* (l'analogia deriva dal mondo sportivo). Graficamente, una pool può essere aperta o chiusa, e solo nel primo caso le lane sono visibili.

- **Lane:** permettono di raggruppare le attività secondo un criterio di ruolo utente o funzionalità, e sono rappresentate con rettangoli all'interno di una pool (al limite collassati). Flow e sequence objects, oltre agli artifacts, sono tutti contenuti in una lane.

14.6 Artifacts

Figura 35 Artefatti: data object, group e annotation

L’utilizzo degli artefatti è utile per aggiungere ulteriore contenuto informativo nei diagrammi BPMN e migliorarne la chiarezza: ad esempio, i **data objects** permettono di specificare le eventuali risorse coinvolte non solo nell’intero processo, ma anche per ogni singolo attività specificandone input o output tramite le associazioni. Similmente, le annotazioni sono brevi indicazioni testuali da inserire come commenti o precisazioni, in analogia con il mondo UML. In ultimo, il concetto di gruppo, presentato come un rettangolo con il bordo visibile in Figura 35, può essere impiegato per raccoglie attività, senza però fornire alcuna accezione funzionale o relativa il flusso di esecuzione.

Bibliografia

1. A. Bongio, J. van Bruggen, S. Ceri, V. Cristea, P. Dolog, A. Hoffmann, M. Matera, M. Mura, A. Taddeo, X. Zhou, and L. Zoni, "COOPER: Towards a Collaborative Open Environment of Project-Centred Learning," Proc. First European Conf. Technology Enhanced Learning (EC-TEL '06), pp. 561-566, Oct. 2006
2. S. Ceri, M. Matera, A. Raffio, and H. Spoelstra, "Flexible Processes in Project-Centred Learning," Proc. Second European Conf. Technology Enhanced Learning (EC-TEL '07), pp. 463-468, 2007
3. Marco Brambilla, Stefano Butti, Piero Fraternali. *Business Process Modeling and Quick Prototyping with WebRatio BPM*. Proc. of BPM Demonstration Track 2010, Hoboken, USA, September 14-16, 2010, Vol. 615 CEUR-WS.org. online: http://ceur-ws.org/Vol-615.
4. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, USA, 2002
5. Marcello La Rosa, Marlon Dumas, Arthur H. M. ter Hofstede, and Jan Mendling. *Configurable multi-perspective business process models*. Inf. Syst., 36:313-340, April 2011
6. F. Charoy, A. Guabtni, and M. Valdes Faura, "A Dynamic Workflow Management System for Coordination of Cooperative Activities," Proc. First Int'l Workshop Dynamic Process Management (DPM '06), vol. 4103/2006, pp. 205-216, 2006
7. J. Lin, C. Ho, W. Sadiq, and M.W. Orlowska, "Using Workflow Technology to Manage Flexible e-Learning Services," Educational Technology and Soc., vol. 5, no. 4, 2002
8. B. Kiepuszewski, A.H.M. ter Hofstede, and C.J. Bussler, On Structured Workflow Modelling, Springer, 2000
9. J. Bardram, J. Bunde-Pedersen, and M. Soegaard, "*Support for Activity-Based Computing in a Personal Computing Operating System*," Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '06), pp. 211-220, 2006
10. P.J. Kammer, G.A. Bolcer, R.N. Taylor, A.S. Hitomi, and M. Bergman, "Techniques for Supporting Dynamic and Adaptive Workflow," Computer Supported Cooperative Work, vol. 9, nos. 3/4, pp. 269-292, 2000
11. Business Process Modeling Notation (BPMN), http://www.bpmn.org/
12. Business Processes Execution Language (BPEL), http://www.oasis-open.org/committees/wsbpel/
13. Web Service Cheorografic Description Language, http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/
14. XML Process Definition Language (XPDL), Workflow Management Coalition, http://www.wfmc.org/standards/xpdl.htm , 2007
15. IBM. IBM BlueWorks Live, 2011. https://www.blueworkslive.com
16. IBM Learning Solutions, IBM, http://www-304.ibm.com/jct03001c/services/learning/ites.wss/zz/en?pageType=page&c=a0001106 , 2007
17. Intalio. Intalio Social BPM, 2011. http://www.intalio.com/social-bpm

18. SoftwareAG. ArisAlign space, 2011. http://www.arisalign.com
19. Oracle. Oracle Business Process Management Suite 11g, 2011. http://www.oracle.com/us/technologies/bpm
20. Fraternali, Brambilla, *A Model-Driven Approach to Social BPM Applications*
21. Elise Olding, Carol Rozwell, and Jim Sinur. *Social bpm: Design by doing*. Technical report, Gartner, May 2010. Gartner research ID Number: G00200281
22. Wil M. P. van der Aalst, Maja Pesic, and Minseok Song. *Beyond process mining: From the past to present and future*. In Barbara Pernici, editor, CAiSE, volume 6051 of Lecture Notes in Computer Science, pages 38{52. Springer, 2010
23. V. Posea, S. Trausan-Matu, and V. Cristea, "Online Evaluation of Students' Opinions about the Collaborative Learning System They Use," Proc. Int'l Conf. Intelligent Computer Comm. and Processing (ICCP), 2007
24. S.W. Sadiq, M.E. Orlowska, and W. Sadiq, "Specification and Validation of Process Constraints for Flexible Workflows," Information Systems, vol. 30, pp. 349-378, 2005
25. S.W. Sadiq, W. Sadiq, and M.E. Orlowska, "Pockets of Flexibility in Workflow Specification," Proc. 20th Int'l Conf. Conceptual Modeling: Conceptual Modeling (ER '01), pp. 513-526, 2001
26. S. Ceri, F. Daniel, M. Matera, and A. Raffio, "Providing flexible process support to project-centered learning," IEEE Trans. Knowl. Data Eng., vol. 21, no. 6, pp. 894–909, 2009
27. F.Casati,S.Ceri,B.Pernici,andG.Pozzi,"*Workflow Evolution*," Data and Knowledge Eng., vol. 24, no. 3, pp. 211-238, Jan. 1998
28. G. Fischer, "*End-user development and meta-design: Foundations for cultures of participation*," in IS-EUD, 2009, pp. 3–14
29. F. Charoy, A. Guabtni, and M. V. Faura, "*A dynamic workflow management system for coordination of cooperative activities*," in Business Process Management Workshops, 2006, pp. 205–216
30. Jeff Howe (June 2006). "The Rise of Crowdsourcing". Wired. http://www.wired.com/wired/archive/14.06/crowds.html Retrieved 2007-03-17
31. Daren C. Brabham. (2008). "*Crowdsourcing as a Model for Problem Solving: An Introduction and Cases*", Convergence: The International Journal of Research into New Media Technologies, 14(1), pp. 75-90
32. Daren C. Brabham. (2008). "*Moving the Crowd at iStockphoto: The Composition of the Crowd and Motivations for Participation in a Crowdsourcing Application*", First Monday, 13(6) http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/2159/1969
33. Daren C. Brabham. (2009). "Crowdsourcing the Public Participation Process for Planning Projects", Planning Theory, 8(3), pp. 242-262
34. Dustdar , Bhattacharya, *The Social Compute Unit*
35. Spopik, Shall, Psaier, Treiber, Dustar, *Towards Social Crowd Environments Using SOA*
36. U.S. Department of Transportation Federal Transit Administration Public Transportation Participation Pilot Program. "*PTP-3 FY 2008 Projects: Crowdsourcing Public Participation in Transit Planning*", http://www.fta.dot.gov/planning/programs/planning_environment_8711.html

37. Amazon Mechanical Turk, https://www.mturk.com/mturk/welcome
38. Missing people search, http://www.wired.com/software/webservices/news/2007/09/distributed_search
39. Armbrust, Fox, Gri-h, Joseph, Katz, Konwinski, Lee, Patterson, Rabkin, Stoica, Zaharia, Above the Clouds: A Berkeley View of Cloud Computing, 2009
40. IBM, "Capturing the Potential of Cloud" 2009
41. IBM, "The Benefits of Cloud Computing" 2009
42. Amazon Web Services, http://aws.amazon.com/
43. Microsoft Windows Azure, http://www.microsoft.com/windowsazure/
44. Mithas, Costello, Tafti, Social Networking in the Enterprise, http://www.computer.org/portal/web/csdl/doi/10.1109/MITP.2011.69
45. Boyd, Danah; Ellison, Nicole (2007). "Social Network Sites: Definition, History, and Scholarship". Journal of Computer-Mediated Communication 13 (1)
46. Ellison, Nicole B.; Steinfield, Charles; Lampe, Cliff (2007). "The benefits of Facebook "friends": Exploring the relationship between college students' use of online social networks and social capital". Journal of Computer-Mediated Communication 12 (4)
47. Kelsey, Todd (2010), Social Networking Spaces: From Facebook to Twitter and Everything In Between, Springer-Verlag, ISBN 9781430225966
48. Casaleggio, Davide (2008). TU SEI RETE. La Rivoluzione del business, del marketing e della politica attraverso le reti sociali. ISBN 88-901826-5-2
49. Arabie, Phipps, and Yoram Wind. "Marketing and Social Networks". In Stanley Wasserman and Joseph Galaskiewicz, Advances in Social Network Analysis: Research in the Social and Behavioral Sciences. Thousand Oaks, Calif.: Sage Publications, 1994, pp. 254–273. ISBN 0-8039-4302-4
50. David Rosenblum (2007). "What Anyone Can Know: The Privacy Risks of Social Networking Sites"
51. Susan B. Barnes (2006-09-04). "A privacy paradox: Social networking in the United States",http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/1394/1312
52. Müller-Prothmann, Tobias (2006): *Leveraging Knowledge Communication for Innovation. Framework, Methods and Applications of Social Network Analysis in Research and Development*, Frankfurt a. M. et al.: Peter Lang, ISBN 0-8204-9889-0
53. GrOWL: A tool for visualization and editing of OWL ontologies, http://www.sciencedirect.com/science/article/pii/S1570826807000157
54. Dependency Inversion Principle, http://www.objectmentor.com/publications/dip.pdf
55. Gamma, Helm, Johnson & Vlissides (1994). Design Patterns (the Gang of Four book). Addison-Wesley. ISBN 0-201-63361-2
56. Wolfgang Pree. *Design Patterns for Object-Oriented Software Development*
57. Dependency Injection, http://www.martinfowler.com/articles/injection.html#FormsOfDependencyInjection
58. AutoFac, An IoC Container for .NET, http://code.google.com/p/autofac/
59. AutoFac, Structuring with Modules, http://code.google.com/p/autofac/wiki/StructuringWithModules

60. Reflection in computer programming, http://en.wikipedia.org/wiki/Reflection_(computer_programming)
61. Dynamic assembly loading in .NET, http://msdn.microsoft.com/en-us/library/system.reflection.assembly.loadfrom.aspx
62. OAuth authenticaction and integrity protocolo, http://oauth.net/