**TECNOLOGIE A LIVELLO DI PIATTAFORMA**

**Architetture di integrazione**

Un sistema informativo è composto da vari moduli che implementano le varie funzionalità richieste. I vari moduli applicativi devono comunicare fra loro, scambiando messaggi e segnalando eventi particolari.   
Tuttavia, non è detto che i moduli che si occupano di svolgere un’attività o un insieme di attività correlate siano omogenei fra loro. Per questo, è necessario trovare un modo per garantire uno scambio di informazioni fra moduli non eterogenei. Le architetture di integrazione risolvono questo problema, garantendo il coordinamento delle attività. Oggetto dell’integrazione sono sia i dati sia i processi usati da vari moduli e che coinvolgono diverse attività.

Integrazione dei dati

I dati gestiti nei database dell’organizzazione, sia nei sistemi OLTP sia in quelli OLAP, i dati possono essere eterogenei per formato e per semantica.

I problemi di integrazione per il formato riguardano:

* La rappresentazione, cioè una stessa informazione può essere rappresentata in modo diverso (ad esempio avere un tipo diverso, cioè uno stesso dato può essere immagazzinato, per esempio, come string in un modulo e come struct in un altro);
* Struttura: un dato formato da più campi può avere i campi ordinati in modo diverso in moduli diversi;
* Presentazione: uno stesso dato può essere presentato con formati diversi, ad esempio un array che contiene lo stesso dato può avere lunghezza massima diversa in moduli diversi.

La semantica, invece, si riferisce al valore attribuito all’informazione. Ad esempio, un “prezzo” può essere inteso nel modulo di un’azienda in euro, nel modulo di un’altra che deve comunicare con la prima in dollari.

Integrazione dei processi

Considerati i processi che coinvolgono più moduli, è molto importante che i moduli comunichino con messaggi che siano ben sincronizzati e abbiano un ordine ben definito. Se questo non accade, si può presentare una situazione di deadlock.

Piattaforme di integrazione  
Per far sì che le varie applicazioni, che gestiscono i dati in modo diverso dal punto di vista della semantica e della struttura, come abbiamo visto, possano comunicare correttamente, si usano delle piattaforme/architetture di integrazione. Esistono varie architetture di integrazione diverse che possono essere utilizzate:

1. *Architettura di integrazione punto a punto*: consiste nel realizzare un’interfaccia diretta fra le applicazioni che devono comunicare. L’interfaccia prende i dati dalle due applicazioni e li trasforma, in modo che siano compatibili fra loro. Lo svantaggio di questo sistema è che rende necessario implementare moltissime interfacce, nello specifico un’interfaccia per ogni coppia di applicazioni che devono comunicare fra loro (e se un’applicazione comunica con molte altre applicazioni, il numero di interfacce diventa molto grande). Inoltre questa soluzione non è scalabile, perché se si modifica un’applicazione si devono modificare anche tutte le interfacce che essa usa.
2. *Architettura di integrazione hub and spoke*: comprende un hub centrale, a cui vengono inviati tutti i dati e i messaggi; esso è collegato a degli spoke, che fungono da adattatori, che adattano il messaggio e lo inoltrano allo spoke o all’applicazione destinataria. In questo modo bisogna implementare una sola interfaccia per ogni applicazione.

Workflow Component   
In sistemi con molte applicazioni, è utile e necessario gestire i flussi di informazione fra le varie applicazioni in modo coordinato e automatico.   
Un Business Process Management System (BPMS), detto anche Workflow Component, è un sistema che consente di gestire i flussi informativi, necessari per un processo, fra applicazioni in modo automatizzato. Esso è un sistema a sé, non è implementato nelle singole applicazioni. Quindi se si deve modificare un processo basta modificare il BPMS per gestire le nuove regole del processo, non si deve modificare direttamente le varie applicazioni.   
Se si devono integrare vari sistemi presi da produttori diversi, si può usare un Workflow Management System, un componente esterno che gestisce la logica dello scambio di messaggi fra sistemi. Esso è separato dai componenti dei sistemi ed esegue una logica di integrazione definita a priori attraverso un modello di processo (che può essere ad esempio uno schema BPMN).

**Modello a servizi**

Esiste un modello di progettazione software alternativo a quello che abbiamo studiato fino ad ora. Esso è realizzato tramite architetture orientate ai servizi (SOA). In questo modello, le funzionalità sono realizzate sempre in modo modulare, ma attraverso dei servizi che possono essere interrogati da remoto attraverso internet. I servizi sono definiti come “task computazionali, con alto livello di disaccoppiamento, che comunicano attraverso la rete”.   
Ogni servizio deve fornire una funzionalità che sia indipendente dagli altri servizi su cui si relazione. Inoltre, deve essere implementato in modo black box per gli utenti, cioè gli utenti devono avere accesso solo all’interfaccia, senza conoscere quello che “c’è dietro”. L’interfaccia definisce un set di operazioni che gli utenti possono effettuare attraverso il servizio.   
Per garantire che le funzionalità fornite dal servizio siano quelle desiderate, esiste un Service Level Agreement (SLA) che definisce le caratteristiche garantite dal servizio in termini di *Quality of Service*. La Quality of Service definisce dei parametri con cui è possibile valutare se il servizio “si sta comportando come desiderato. Questi parametri possono essere il tempo di risposta, il costo, …

Le informazioni su un servizio sono salvate in un Service Registry: esso memorizza i protocolli che si possono usare, i messaggi che possono essere scambiati, le funzionalità offerte dal servizio, le modalità con cui si può accedere al servizio.   
La SOA può usare per comunicare tramite internet alcuni protocolli definiti sopra HTTP, usando la codifica XML (questa scelta è detta procollo SOAP), oppure con interrogazioni dirette in HTTP. Questo secondo caso viene incluso nello stile “REST”, (REpresentational State Transfer), che supporta le interrogazioni HTTP (GET, POST, …) e fornisce risposte in formiti HTML, JSON, …

Spesso i servizi non sono utilizzati singolarmente, ma in modo integrato: tanti servizi collaborano per fornire una funzionalità (ad esempio una funzionalità di e-shopping avrà un servizio per gestire il carrello, un servizio per il pagamento, …) . Tutti i servizi che implementano una funzionalità compongono un servizio più grande e complesso. Per gestire correttamente un servizio composto da più servizi, serve definire una *service orchestration*, cioè definire la sequenza con cui si possono invocare i vari servizi.   
È conveniente che la service orchestration sia dinamica, cioè un servizio completo viene creato selezionando in modo dinamico, *al momento*, i vari servizi che andranno a comporlo, fra tutti quelli disponibili.

Il paradigma a servizi può essere usato come un mezzo di integrazione: esso infatti definisce a priori (nel *Service Registry*) un pool di possibili messaggi da scambiare. Quindi le applicazioni di integrazione possono essere costruite basandosi su questo paradigma.

Un esempio di questo approccio è l’utilizzo di un *Enterprise Service Bus*. Un sistema a Enterprise Service Bus utlizza un’infrastruttura di integrazione (Il Service Bus, per l’appunto) basata sui servizi. Tramite questo Bus è possibile usare un service registry che tenga traccia di tutti i servizi da utilizzare e da integrare. Tutte le applicazioni hanno accesso a questo unico bus di integrazione.