Parliamo di SOA (Service Oriented Architecture)

Antonio Pintus, Marco Marongiu

Chi siamo

Antonio Pintus è laureato in Informatica e studente di Dottorato di Ricerca in Informatica con argomenti relativi a "Service Oriented Architecture". Lavora come Software Engineer presso il CRS4 (Centro di Ricerca, Sviluppo e Studi Superiori in Sardegna). pintux[AT]pintux.it

Marco Marongiu è laureato in Matematica. Lavora come System Administrator per Tiscali nella sede di Sa Illetta a Cagliari. Attualmente il suo compito principale è la gestione sistemistica dell'infrastruttura SOA di Tiscali, basata su prodotti TIBCO. Scrive inoltre su riviste e siti web specializzati, in Italia e all'estero. mmarongiu[AT]tiscali.com

Di cosa parliamo...

- Sistemi eterogenei, non progettati per parlare fra loro, da integrare, di qualunque dimensione
- Sistemi distribuiti di grandi dimensioni, fuori dal controllo di una singola organizzazione
- Sistemi in ambito “business”, ma ultimamente la SOA si sta utilizzando anche in ambito scientifico

Service Oriented Architecture (SOA): introduzione (1)

- In un recente passato, i sistemi enterprise, spesso si differenziavano (di fatto creando notevoli incompatibilità tra di essi) per l'adozione di tutta una serie di diversi protocolli, interfacce, tecnologie e prodotti commerciali

- Nonostante ci siano stati tutta una serie di approcci per rendere più interoperabili questi sistemi, è sinora mancato una effettiva definizione ed impiego di standard

Service Oriented Architecture (SOA): introduzione (2)

- Inizialmente nata come soluzione software per l'integrazione di sistemi eterogenei in ambito Business-to-Business (B2B)
- Giunta a poter essere considerata, da un punto di vista più astratto, come un'architettura, un modello di programmazione ed un paradigma con lo scopo di raggiungere un disacoppiamento (o basso livello di accoppiamento) nel design di applicazioni distribuite costituite da agenti software (servizi) interagenti tra loro
- Modello pervasivo, oltre che inter-enterprise, anche intra-enterprise e a tutti i livelli, dal management al settore IT

Vantaggi di SOA

- Interoperabilità tra sistemi eterogenei
- Standardizzazione delle interazioni
- Astrazione dai dettagli implementativi
- Scalabilità
- Adattività ai rapidi cambiamenti delle esigenze di business
- Monitoring
- Governance
- Libertà nella scelta delle piattaforme di sviluppo

Svantaggi di SOA

- l'aggiunta di un certo numero di livelli di indirezione porta ad una maggiore lentezza
- Complessità: sviluppare, fare il debugging e mantenere sistemi distribuiti con un basso livello di accoppiamento tra le componenti è piuttosto complesso
- Spesso l'adozione di questo modello impone un adattamento anche nell'organizzazione delle persone

Concetti base

- Servizi
  - insieme di componenti software che implementano una funzionalità
- Interoperabilità
  - capacità di sistemi diversi di comunicare fra loro
- Disaccoppiamento (loose coupling)
  - condizione di dipendenza ridotta o assente fra diversi componenti di un sistema distribuito

Componenti base

- Infrastruttura
  - Comunemente è l'Enterprise Service Bus (ESB), il cui ruolo è permettere l'interoperabilità
- Architettura
  - ad alto livello: i sistemi, i servizi, le interfacce, il bus
- Processi
  - Sequenza di azioni da eseguire per implementare una certa funzionalità di business (ne parleremo in concreto più avanti)

Il concetto di Servizio (1)

- Un servizio è un modulo software, autonomo e indipendente dalla piattaforma, indipendente dal linguaggio di programmazione usato e progettato per consentire una interazione di tipo Machine-to-Machine (M2M) attraverso la rete.
- E' un'astrazione software di un processo reale
  - In particolare potrebbe essere un servizio orientato al Business-to-Business (B2B)
- Fornitore del servizio (Service **Provider**): la parte che fornisce e rende disponibile il servizio.
- Fruitore del servizio (Service **Consumer**): la parte che consuma il servizio utilizzandolo per il proprio processo di business

Il concetto di Servizio (2)

- I servizi diventano quindi le unità “atomiche” per il design e l'implementazione di architetture software distribuite che rispondono all'esigenza di processi di business anche molto complessi nonché potenzialmente decentralizzati ed eterogenei.
- I servizi possono essere composti tra loro
  - Nuove applicazioni possono essere “rapidamente” costruite mediante una loro composizione
- Si parla di **orchestrazione** e **coreografia** di servizi
  - esistono approcci alternativi; fra questi stanno prendendo quota gli **eventi**, da cui le Event-Driven Architecture
- Le applicazioni composte sono a loro volta un servizio

Interoperabilità e disacoppiamento (loose-coupling) (1)

- Web Services: la tecnologia dei Web Service è rapidamente diventata la tecnologia abilitante per SOA.
- Web Services (WS), una definizione:
  - *“A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.” (W3C)*
- Grazie alle descrizioni formali e gli standard definiti per i WS, essi sono oggi ampiamente utilizzati, riuscendo a garantire anche l'interoperabilità, il disacoppiamento e indipendenza dalla piattaforma.

Interoperabilità e disacoppiamento (loose-coupling) (2)

- I web service sono un sistema **sincrono**
- La comunicazione può essere anche **asincrona**
  - **Nessuno dei due schemi è vincente di per sé, la scelta deve essere strettamente legata al contesto in cui il servizio andrà ad operare**
- Un approccio asincrono è basato sullo scambio di messaggi, che può avvenire secondo modelli diversi (si parla di Message Exchange Patterns o MEP)
- Uno strumento è fornito da prodotti che implementano la specifica JMS (messaggistica basata su code o su “topic”)

Enterprise Service Bus (ESB)

- Realizza la comunicazione fra i servizi
- Permette l'invocazione dei servizi su sistemi eterogenei
  - supporto allo scambio dei messaggi
  - la trasformazione dei dati
  - il routing intelligente dei messaggi
  - gestione dell'affidabilità e sicurezza
  - monitoring e management dei servizi
- Costituisce una “backbone” che tratta le applicazioni come servizi, consentendo la composizione e la comunicazione tra applicazioni con: loose coupling, Messaggi, “Plug & Play” di applicazioni enterprise eterogenee

Enterprise Service Bus (ESB)

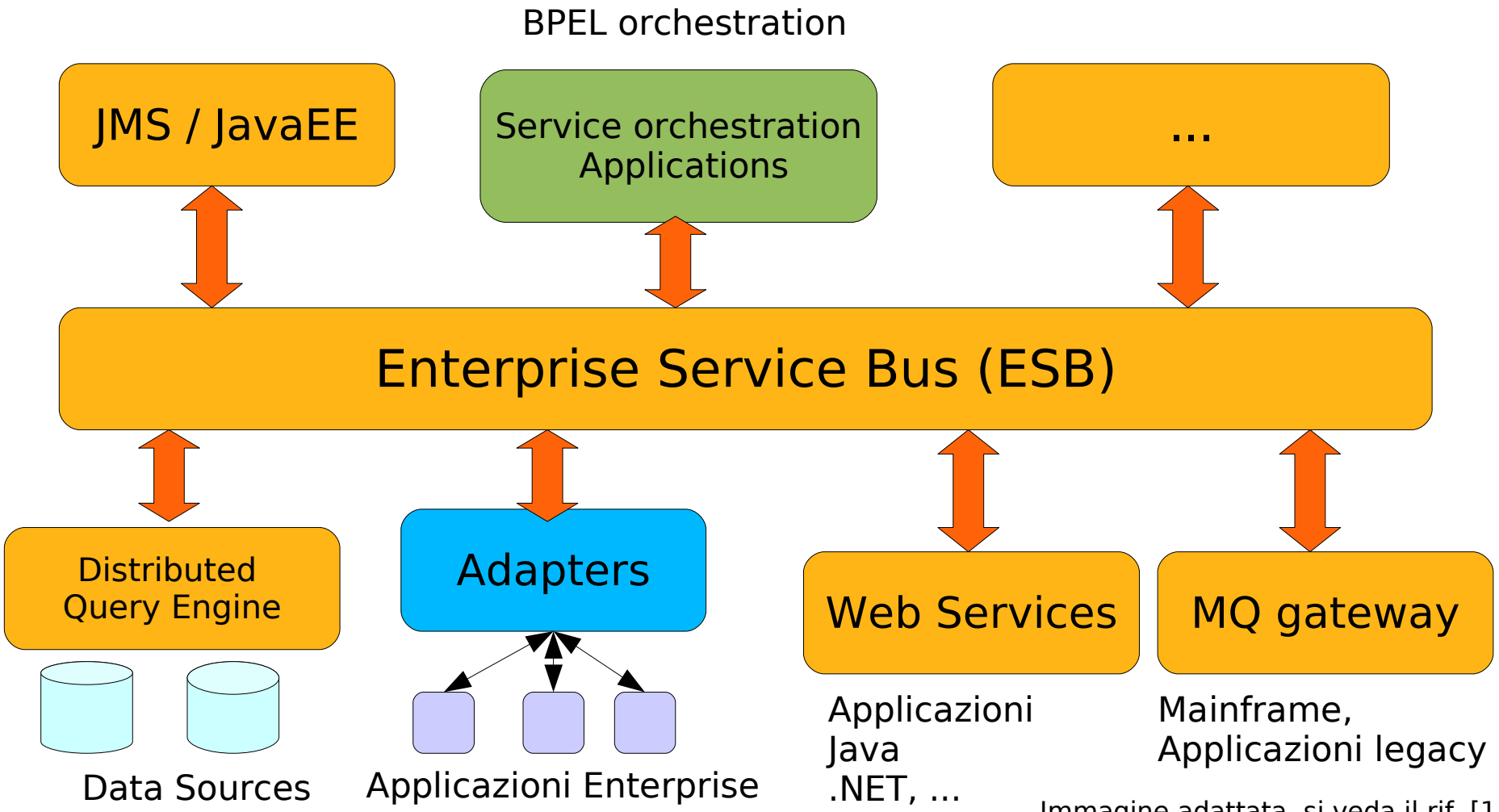


Immagine adattata, si veda il rif. [1]

Service Orchestration

- Un'orchestrazione di servizi descrive come essi possono interagire tra di loro a livello di messaggi, includendo la logica di business e l'ordine di esecuzione delle interazioni
  - L'intero flusso e la sua esecuzione sono sempre controllati da una delle parti coinvolte nel processo.
- Mediante orchestration quindi è possibile definire un processo di business multi-step, eseguibile, potenzialmente di lunga durata e transazionale
  - WS-BPEL il linguaggio più usato

Service Coreography

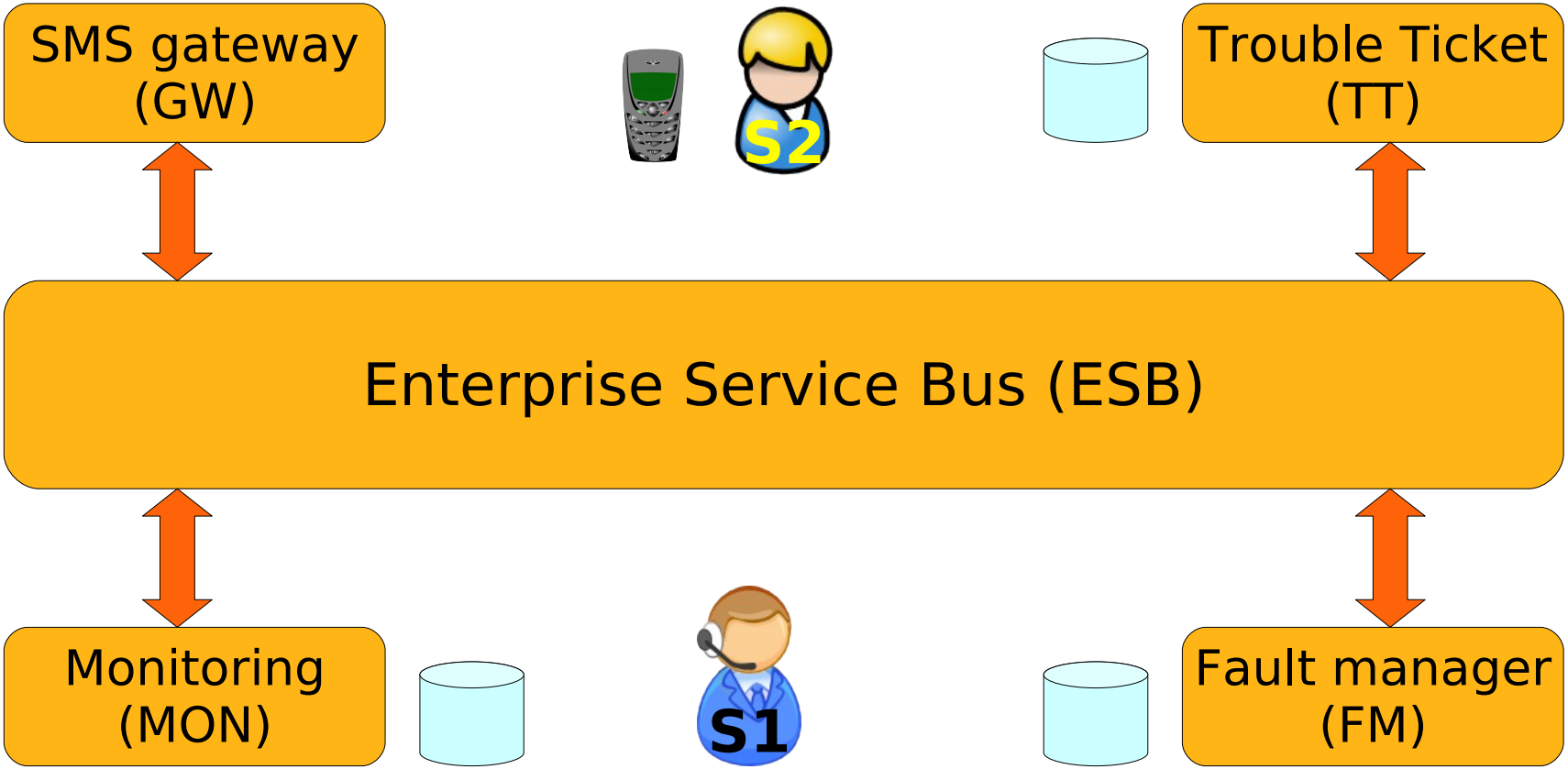
- Ogni servizio conosce solo il suo “compito” e come passare ai servizi che lo seguono nella catena il prodotto della loro elaborazione
  - è il modello del “passacarte”
  - Non c'è una “entità centrale” che ha conoscenza di tutto il processo e lo dirige
- Vantaggi: l'implementazione di un servizio coreografico è concettualmente più semplice
- Svantaggi:
  - capire dove un processo si è “incastrato” può diventare complicato
  - non ci sono standard e tool consolidati come per BPEL e orchestrazione

API

- In questo caso si utilizzano i prodotti di un vendor che mette a disposizione una API e dei tool di sviluppo per costruire i servizi
- I servizi e il bus condividono la API
- Vantaggi: lo sviluppatore non deve occuparsi di dettagli di basso livello (p.e.: protocollo di comunicazione, reliability, gestione degli errori...)
- Svantaggi: si è legati al vendor, ai suoi prodotti e alle loro modifiche, al suo servizio di supporto...

Esempi

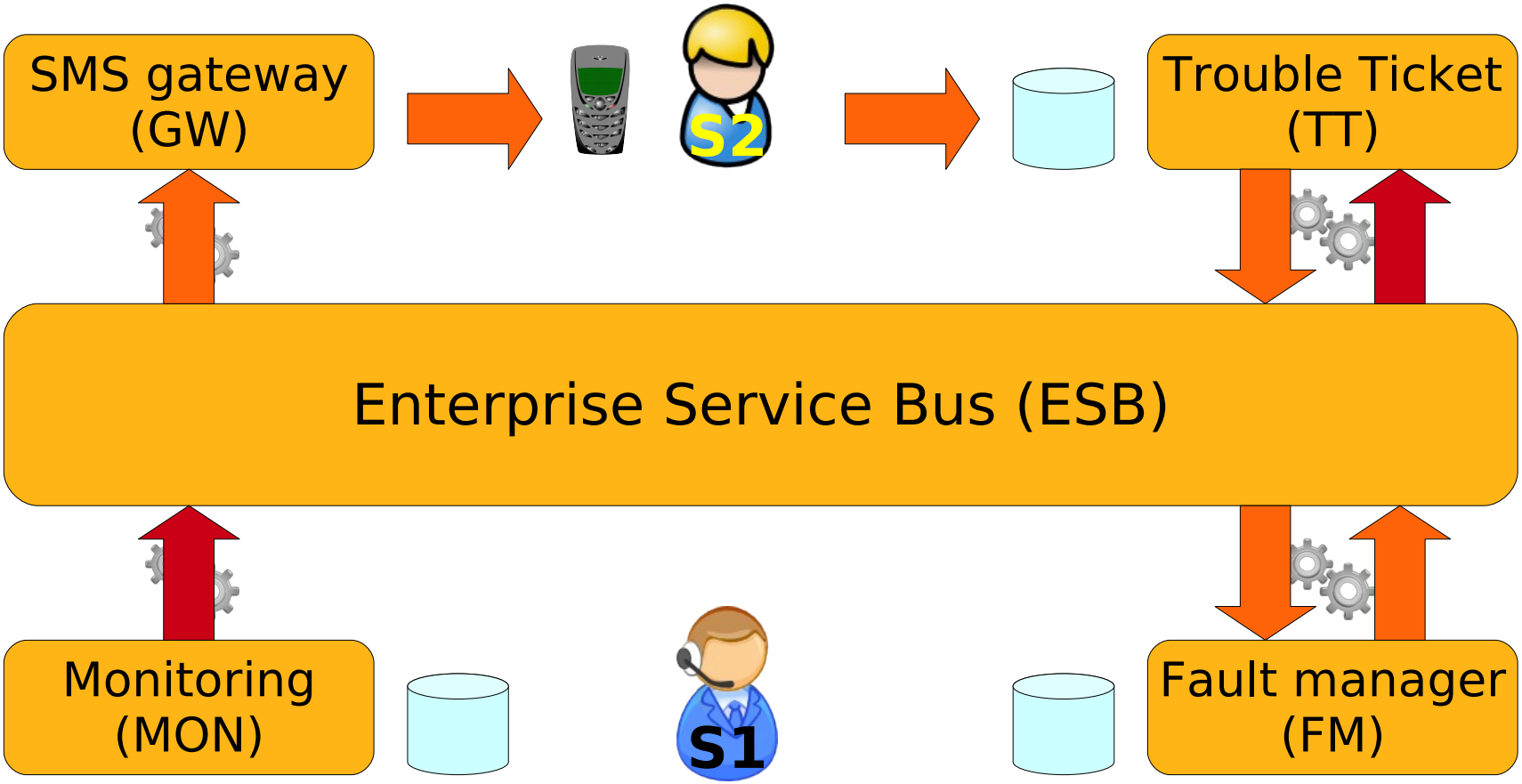
Architettura di esempio



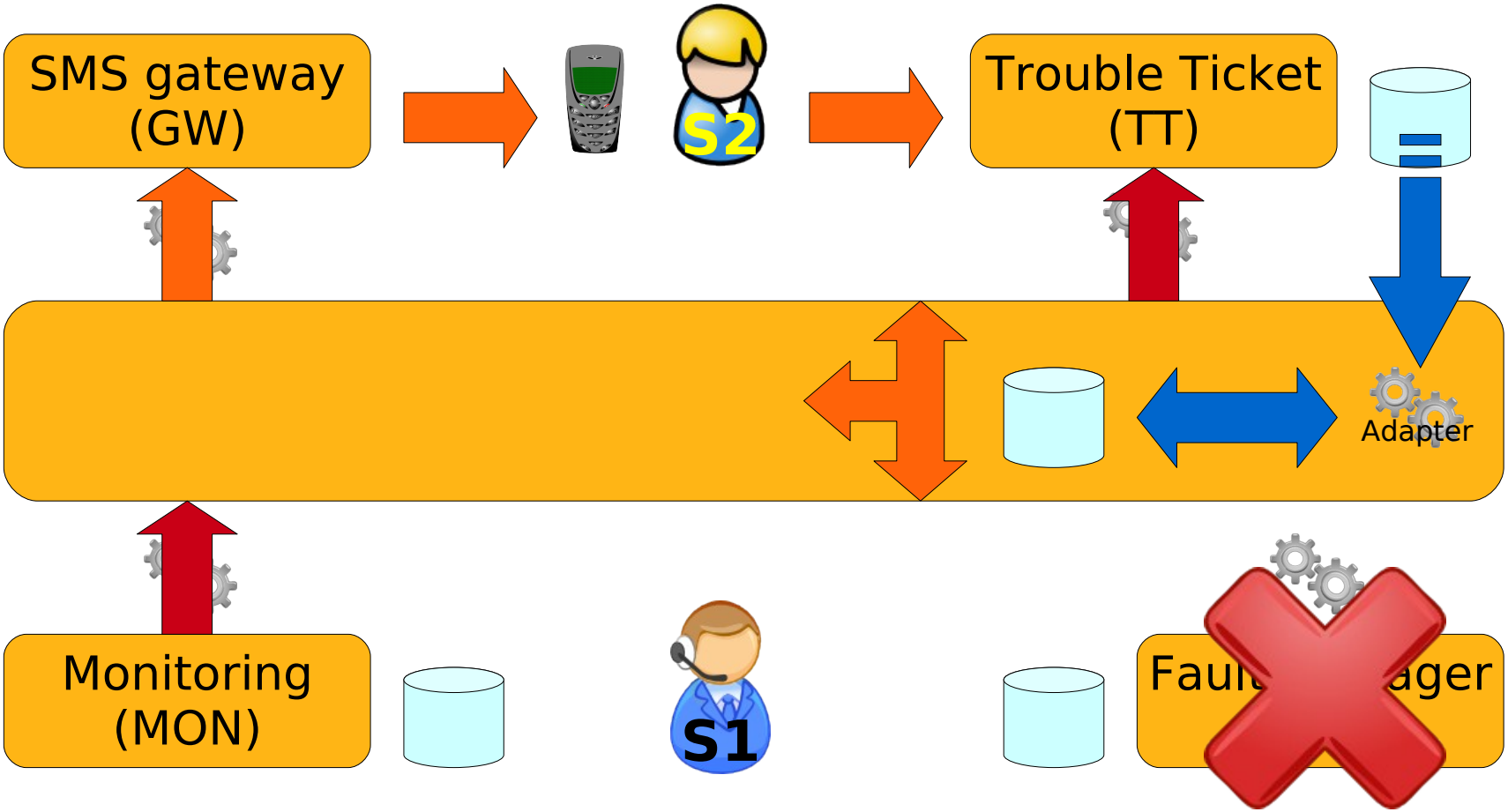
Esempio: flusso non integrato

- qualcuno inserisce un ticket di fault su TT
- S1 vede il fault, lo prende in carico e allerta S2
- S2 importa il fault su FM
- FM invia per SMS attraverso GW gli aggiornamenti del ticket
- risolto il problema, S2 chiude il ticket su TT
- FM rileva la chiusura del fault e manda l'ultimo aggiornamento via GW

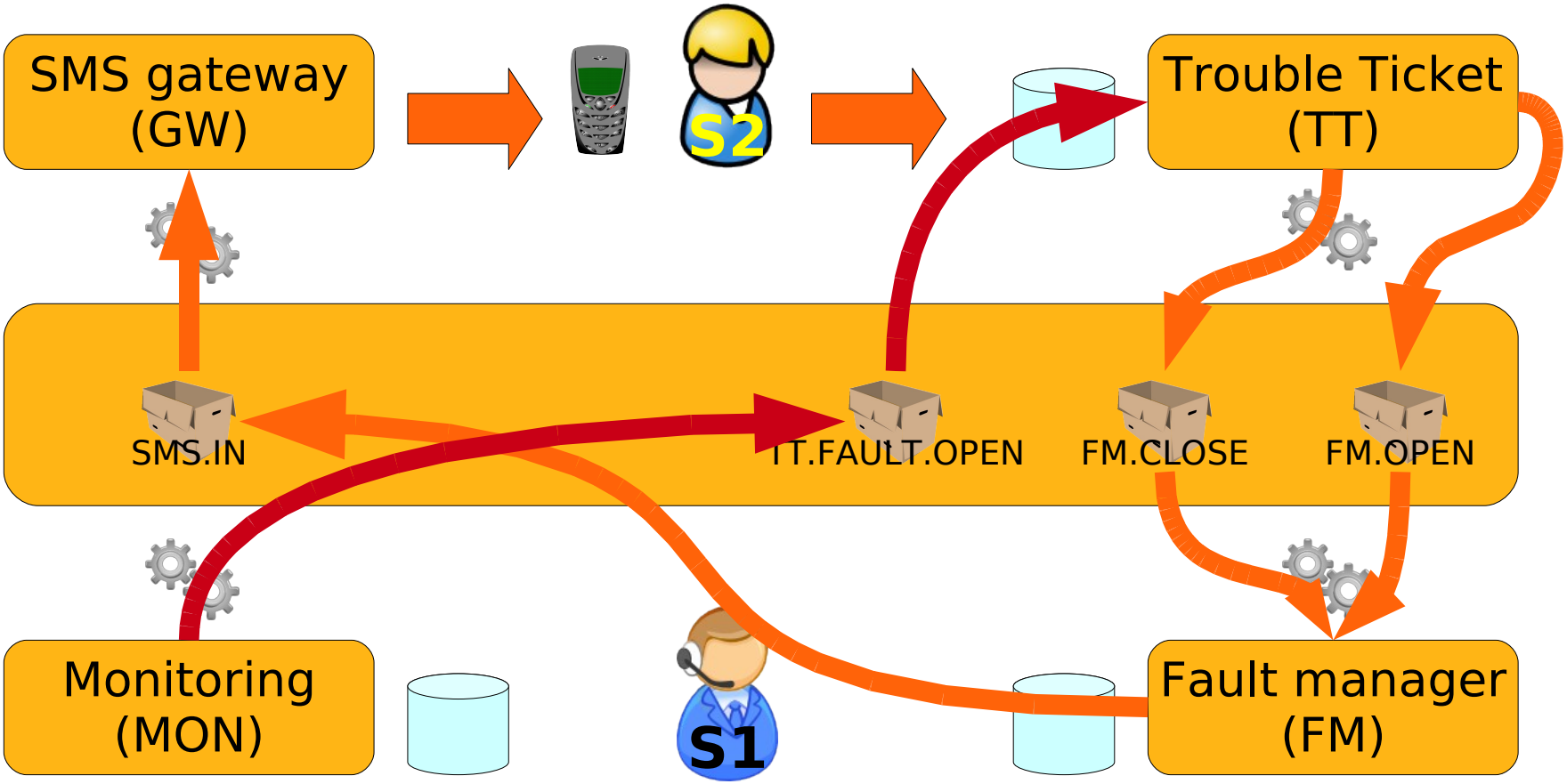
Coreografia (con messaggi one-way)



Coreografia (ottimizzato)



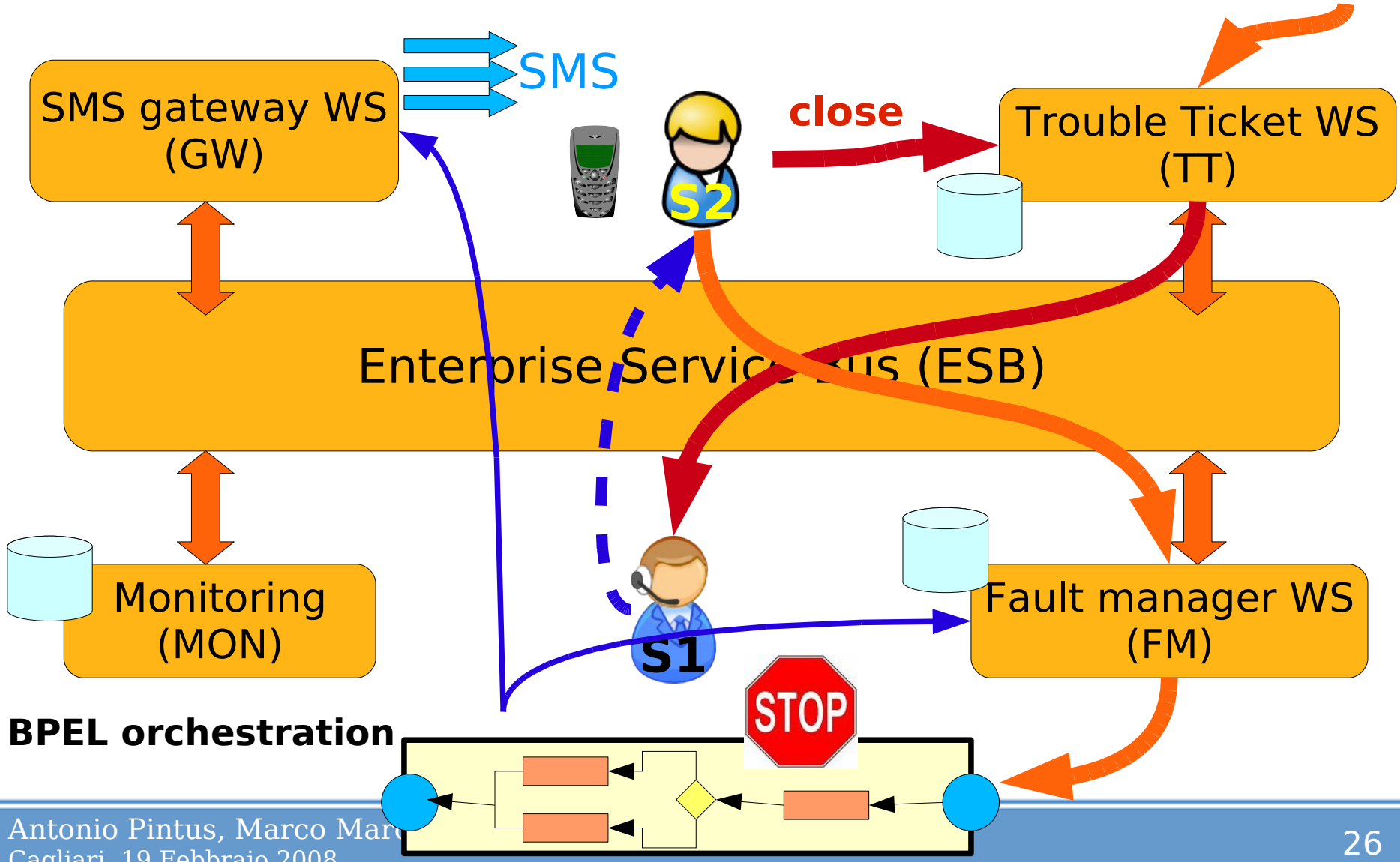
Messaggi asincroni



Un altro approccio: Web Services (1)

- Tutti i servizi sono realizzati:
  - O come Web Service
  - O dispongono di un adapter di tipo Web Service
- La gestione dei processi automatici ora avviene mediante una Web Service Orchestration:
  - Mediante un processo BPEL, “long-running”, capace di reagire ad eventi (asincroni)
  - Per esempio alla chiusura del ticket che termina di fatto il processo

Un altro approccio: Web Services (2)



Riferimenti

[1] Michael P. Papazoglou, *“Web Services: Principles and Technology”*, Pearson Prentice Hall, 2007

[2] Nicolai M. Josuttis, *“SOA in Practice – The Art of Distributed System Design”, O'Reilly, 2007*

[3] AA.VV.: *“SOA Practiotioners' Guide”, part 1, 2, 3.*
http://dev2dev.bea.com/pub/a/2006/09/soa-practitioners-guide.html
http://www.soablueprint.com/practitioners_guide

Grazie per l'attenzione.