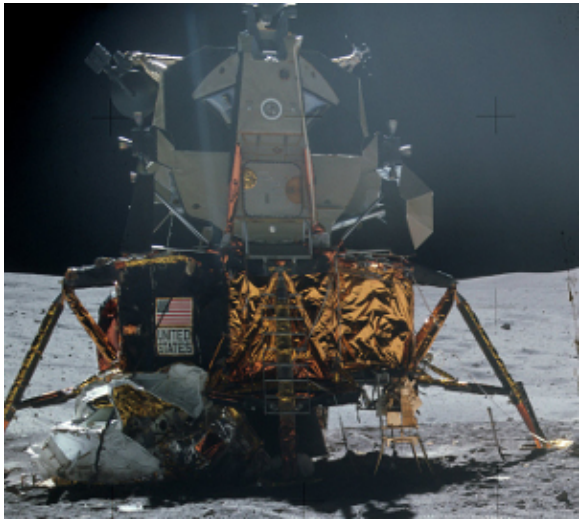
Embracing Modularity

Liferay 7 - OSGi

Mauro Mariuzzo, Software Architect – SMC Treviso

Cosa pensi se dico “Modulo”?



Moduli nel software

Nel software un modulo è

> *Una porzione di programma che svolge una specifica funzionalità. Porzione che può essere usata da sola o combinata con altri moduli del medesimo programma*

Quindi “modulo” identifica qualcosa che implica incapsulamento e riutilizzo

- Se mi serve una CPU non devo negoziare con Intel il ciclo di progettazione e produzione
- Posso usare una CPU AMD se predisposta per il medesimo socket

Perchè abbiamo bisogno di moduli?

Un Boeing 747-400 contiene:

- 6.000.000 componenti
- 274km di cavi
- 8km di tubi

Il suo progetto è composto da ~75000 disegni tecnici... realizzati da migliaia di tecnici... 50 anni fa

La necessità di moduli nel software?

Un aereo è complesso, ma anche il software non scherza!

NASA Space Shuttle:	500.000 linee di codice
Lettore DVD:	1.000.000 LoC
Windows XP:	45.000.000 LoC
Windows 7:	40.000.000 LoC
BMW serie 5:	50 computer interconnessi
Google:	2.000.000.000 LoC

Grazie; ma io sto già usando i moduli :)

I sistemi di versionamento (SVN, Git) consentono di strutturare i repositori “a modulo”

Gli strumenti di build (Ant+Ivy, Maven) gestiscono il concetto di modulo

Maven chiama i suoi progetti moduli:

- I moduli hanno un preciso nome e numero di versione
- Il nome del modulo è indipendente dal nome del file. Molti utenti e tool lo dimenticano :(
- Se un progetto non si trova su maven central allora non esiste :(

Anzi, usavo i moduli già da bambino :)

Lego è il gioco più ingegnoso e democratico al mondo. Con i suoi mattoncini siamo in grado di costruire tutto!

Il problema “IKEA”

Compri una scatola di tubi, pannelli, viti, tasselli... e speri di ottenere una libreria

- Le istruzioni sono in Russo o Svedese
- Mancano due viti e tre tasselli...

Risolvere il problema “IKEA”

Forza bruta!

Con abbastanza tempo e denaro...

tutto è possibile!

Qualcuno ha realizzato una casa fatta interamente con i mattoncini lego

- Ci ha messo 2 settimane
- Ci sono volute 1200 persone
- Sono stati usati 3,3 milioni di mattoncini

Ma i moduli non sono abbastanza

Maven Central è un enorme contenitore di Lego...

...senza istruzioni

Creare o gestire progetti di una certa dimensione può risultare davvero difficile e complesso

Serve una modalità di ricerca efficace

il nome non basta, è limitante

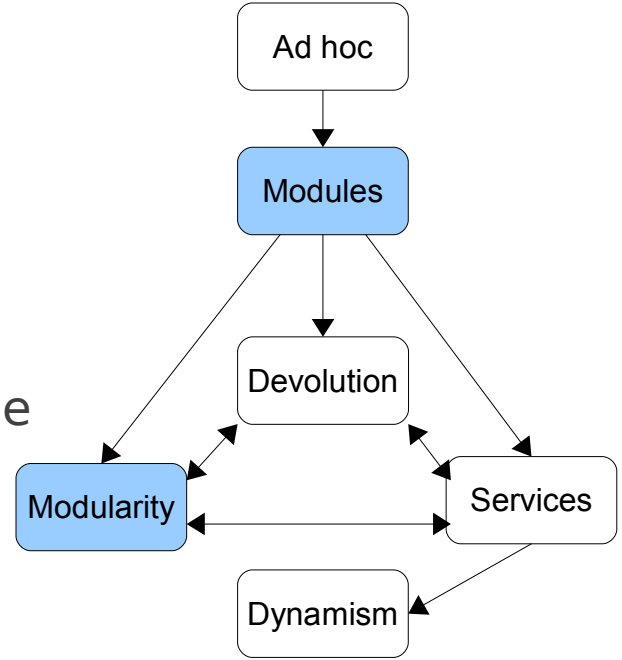
La Modularità

Moduli e Modularità non sono la stessa cosa!

Solo Nome non basta per la Modularità

I veri Moduli possiedono un “contratto”

Il Contratto definisce le aspettative di entrambe le parti



La vera Modularità

La modularità richiede che il modulo si “auto-descriva”

Deve elencare i suoi “requisiti”, e le sue “capacità”

Ad esempio un file .jar dovrebbe:

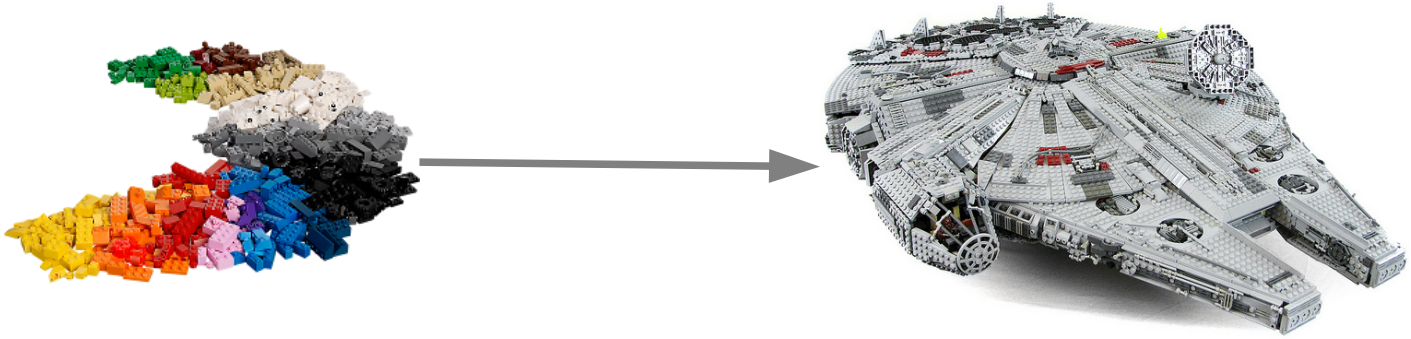
- Indicare il suo nome e la sua versione
- Indicare in modo esplicito le dipendenze con il numero di versione
- Indicare in modo esplicito le funzionalità che esporta indicando la versione per ognuna

Congratulazioni - abbiamo realizzato un bundle OSGi!

Il “manuale di istruzioni” mancante

Dipendenze e funzionalità, con relative versioni, forniscono ad ogni modulo il suo manuale.

Il componente “conosce” dove si può/deve posizionare nel progetto finale



Benefici della Modularità

Assemblaggio automatico del “componente da eseguire”
    Basta ClassNotFoundException

Vero disaccoppiamento dall'implementazione
    Dichiaro cosa mi serve, non dove si trova

Basta assunzioni sui componenti
    ~~((FooImpl)foo).internalMethod();~~

Garantirsi un futuro

Assegnare un numero di versione alle funzionalità esportate dà maggior vita al contratto

1.x	→ 2.0	:= cambiamenti critici alle API
1.x	→ 1.y	:= cambiamenti retro-compatibili (ma critici per un provider)
1.x.y	→ 1.x.z	:= bug-fix (tutti felici)

Chi utilizza le API può tranquillamente accettare un intervallo di versioni

Tutto ciò è chiamato “semantic versioning”

Benefici del Versionamento Semantico

E' un po' come prevedere il futuro!

E' facile capire quando le modifiche apportate ad un modulo impediranno al mio di funzionare

Ci sono strumenti che aiutano la gestione semantica del mio codice

- Bnd/Bndtools in Eclipse
- Apache Aries plugin

Ottengo un errore in compila se dimentico di aggiornare le versioni

Disaccoppiare

E' appagante realizzare moduli consistenti
    Ma come si realizza la condivisione?

L'operatore “new” o classi factory mi fanno tornare in un ambito di relazioni strette

La risposta è Microservices!

Con i Microservices realizzo dipendenze a livello di API
    Componenti disaccoppiati esistono e coesistono in quanto parte dell'architettura

Microservices: così semplice?

I Microservice sono popolari e leggeri

- Ma abbastanza snelli da soddisfare qualsiasi esigenza di integrazione?
- Come faccio a gestire tutte le configurazioni?

OSGi consente in-VM Microservices!

Il framework prevede un registro di servizi (e notifiche)

- Componenti condivido servizi attraverso il registro
- Nessuna configurazione, nessun costo elaborativo

Riassumendo

La modularità è l'unica strada per gestire progetti “grossi”

Nessuno realizza monoliti intenzionalmente: “succede”!

Ci deve essere un forte controllo sul progetto per impedirlo

Sviluppare a moduli è facile usando i tool giusti

Aggiungere modularità a monoliti non è impossibile

- Però non sarà mai il dolce che ti aspettavi

Perché...

Perché parliamo di modularità?

Perché Liferay ha intrapreso il difficile percorso verso la modularità?

Perché noi dovremmo adottare la modularità

- Nel nostro modo di sviluppare?
- Nel nostro modo di analizzare requisiti e funzionalità?
- Nel nostro modo di fare business?

Perché? Velocità!

Passiamo sempre più tempo online

Gli acquisti online sono in continuo aumento

E' necessario reagire tempestivamente ai cambiamenti

Realizzare un nuovo servizio in pochi mesi può non rivelarsi efficace...

... la finestra temporale potrebbe essere poche settimane, o pochi giorni

Perché? Velocità!

Come possiamo bilanciare la volontà di innovazione e di sviluppi veloci con la necessità di mantenere sotto controllo sistemi che diventano via via più grandi e complessi?

Impariamo dai casi di successo

In Natura i sistemi complessi sono gestiti attraverso l'applicazione e la replicazione di schemi semplici

Impariamo dai casi di successo

L'essere umano osservando la Natura ha imparato ad imitarla.

Da piccoli componenti...

Impariamo dai casi di successo

... a soluzioni fantasiose

Microservice per tutto?

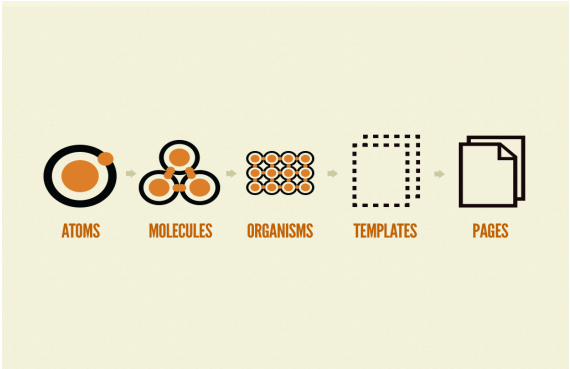
Microservices risolvono solo una parte dell'equazione...

E per la User Interface?

User Interface

La tendenza è di creare linguaggi modulari per descrivere l'interfaccia

Material Design



Atomics Design

Javascript

Anche Javascript sta diventando davvero modulare

~~ECMAScript 6~~

ECMAScript 2015

Aspetti Chiave

1

Consentire **indipendenza** di sviluppo, deploy ed evoluzione

2

Combinare insieme i componenti esistenti

3

Riutilizzo ed agevolare la **personalizzazione**

Modularità

Sfide

Microservices
- Nuovo paradigma di sviluppo
- Come combinare i componenti
- Operazioni complesse

EcmaScript 2015
- Ancora non supportato dai browser

Design Languages
- Alcuni troppo specifici
- Alcuni troppo generici

Liferay 7

Usa i Microservices adesso!

1. In-VM Microservices
2. Registry & Communication: meccanismi collaudati
3. Molteplici opportunità di customizzazione

Liferay 7

Javascript modulare direttamente dal futuro

1. Sviluppa usando EcmaScript 2015 come se fosse già ampiamente diffuso
2. Liferay lo tradurrà automaticamente in quanto supportato oggi

Liferay 7

Riutilizzo e customizzazione di componenti di livello Enterprise

Fornisce migliaia di componenti (moduli) frutto della conversione modulare delle funzionalità già presenti in Liferay. Le potenzialità della piattaforma sono preservate e migliorate

Liferay 7

Innovare è già difficile di suo...

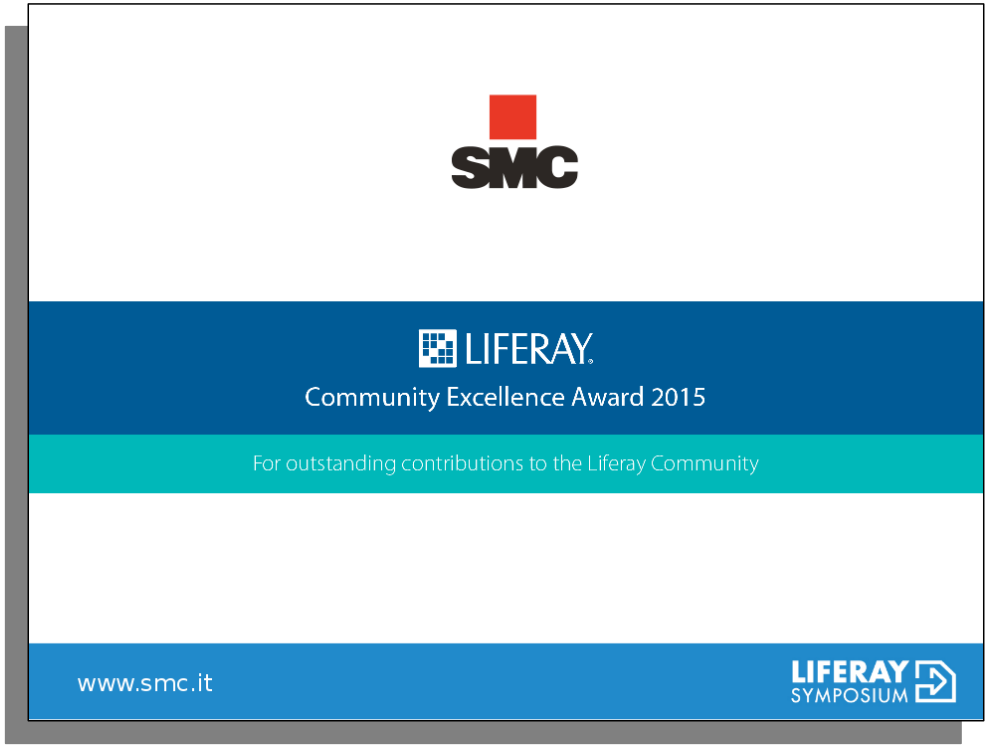
La tecnologia che usi deve essere un tuo alleato!

Embracing Modularity

OSGi e Liferay 7

Mauro Mariuzzo
Software Architect - SMC Treviso

mauro.mariuzzo@smc.it
@MariuzzoMauro



Credits

Devcon 2015

- Modules and Modularity – Tim Ward
- Speed up innovation through modularity – Jorge Ferrer

Immagini da

Wikipedia, .lego.com, .eurobricks.com, .pintrest.com, .appelmo.com, .plunatic.fr, .paolisi.com

Modularity Maturity Model – Graham Charters (IBM)