



SOLUTA.NET

L'Architettura come fattore chiave per un processo di sviluppo Agile

Pierfranco
Ferronato
Chief Architect
Soluta.net

Soluta.Net

*Primo Forum AICA
sulle Tecniche Agili per il
Software Process Improvement
(Milano)*



Licence

This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/1.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

You are free:

to copy, distribute, display, and perform the work

to make derivative works

to make commercial use of the work

Under the following conditions:

Attribution. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

For any reuse or distribution, you must make clear to others the license terms of this work.

Any of these conditions can be waived if you get permission from the author.

About

Soluta.net is constituted by a team of IT professionals that have a world wide experience in Component Based Development and Enterprise Architectures. Soluta Architects have provided technical and architectural leadership for several European projects using advanced Internet-related technologies, component-based development, webservices and wireless technologies in a number of domains, including telecom, pharmaceutical, CRM, EAI and tourism.

The **Soluta.net** approach is to leverage the customer experience and assets while setting up a sound architecture. The experience gained in 15 years of industrial experience with international customers allows Soluta.net to make use of the best breed of technologies, open standars, tools and methodologies. Soluta.net is focused on Open Source and Open Standards.

Pierfranco Ferronato is the Chief Architect and founder of Soluta.net. He has over 15 years of experience in all aspects of distributed systems development and is internationally recognized as an expert in large-scale architectures and object-oriented/component development. Dr. Ferronato has provided technical and architectural leadership for several European projects using advanced Internet-related technologies, component-based development, webservices and wireless technologies in a number of domains, including telecoms, pharmaceutical, CRM, EAI and tourism. He is an active member of the OMG and a frequent speaker at conferences worldwide.



La frustrazione

- ▶ I progressi nelle tecniche di software engineering e di project management non reggono al passo alle esigenze di business
- ▶ Sviluppare Software è
 - ▶ Complesso
 - ▶ Propenso agli errori
 - ▶ Altamente dinamico ... frustrante?
- ▶ Il processo di sviluppo è ad alto rischio e con esso ogni progetto
- ▶ Nel 1995, solo il 16% dei progetti software è stato terminato in tempo e nei costi previsti⁽¹⁾



Processi di sviluppo in retrospettiva

- ▶ '70 nessuna regola: *i pionieri*
 - ▶ Nel 1968 Edsger Dijkstra scrisse la lettera “GOTO Statement Considered Harmful”.
- ▶ '80 progetti complessi, nuove opportunità di business, tempi lunghi e domini più vasti: *ingegnerizzazione ed ordine*
- ▶ '00 tempi stretti, costi minimi. settori vasti, tecnologia variegata e complessa.
 - ▶ Le regole diventano troppo strette, vi è un movimento di reazione agli approcci pesanti (XP): *la reazione*
 - ▶ *Agilità è un fattore relativistico*
- ▶ '??, agilità come maturità

Architettura in retrospettiva

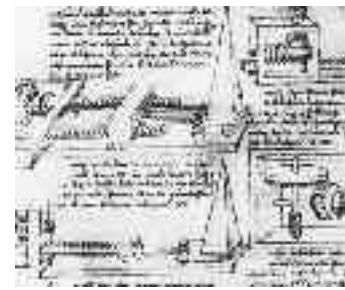
- ▶ '70: host, sistemi monolitici
 - ▶ Architettura: cos'è?
- ▶ '80: Personal Computers, Local Area Networks, two tiers, riuso “development time”
 - ▶ Ingegnerizzazione dell'Architettura



Architettura ed ingegneria



- ▶ Ingegnerizzazione sì, ma...
- ▶ confrontare software con l'ingegneria civile non è corretto
 - ▶ nessuno chiede che un ponte venga allungato o un edificio spostato
- ▶ Non si deve cementare il software con un approccio che non gli è congegnale
 - ▶ Crea un falso senso di sicurezza: *la colpa è poi di chi non ha saputo seguire il processo*
- ▶ L'ingegneria aerospaziale ha da tempo abbandonato i metodi dell'ingegneria aeronautica
- ▶ Si deve trarre dalla flessibilità un punto di forza e non trattarlo come una debolezza ed ingabbiarlo/eliminarlo
 - ▶ Agile Software Foundation



Architettura "consapevole"

- ▶ L'agilità dell'IT non deve essere imbrigliata
 - ▶ Agilità come un punto di forza;
- ▶ Un'architettura flessibile abilita ad un processo agile, mentre un processo agile con un'architettura rigida non porta a tutti i potenziali benefici.
- ▶ Esempi
 - ▶ minimizzare le dipendenze tra semilavorati. Diverse strategie; language class, componenti, applicazioni, federazioni;
 - ▶ separazione tecnologia/funzionalità;
 - ▶ generazione del codice e feedback retroattivo processo-architettura;
 - ▶ principio di Pareto: l'80% dei benefici viene dal 20% delle funzionalità;



Minimizzare le dipendenze

▶ Inversion of Control (intra componente)

```
public class AppleImpl implements Apple{  
    private Orange orange;  
    public Apple() {  
        this.orange = new OrangeImpl();  
    }  
    // other methods  
}
```

```
public class AppleImpl implements Apple{  
    private Orange orange;  
    public setOrange(Orange pOrange) {  
        this.orange = pOrange;  
    }  
    // other methods  
}
```

▶ Smart proxies (inter componenti)

- ▶ Distribuzione run-time dei mediatori ai componenti distribuiti

▶ Semantica

- ▶ Migliorare la comunicazione e comprensione:
- ▶ Applicazioni, componenti, operazioni, parametri, eccezioni, eventi, classi (?)
- ▶ Ontologia, repository, registry



Tecnologia / Funzionalità

▶ Così

```
[omissis]
try{
if (System.getSecurityManager() == null) {
    System.setSecurityManager(new RMISecurityManager());
}
} catch { /*tech Error handling*/}
try {
PowerService service = (PowerService) Naming.lookup("rmi://" + serverIp+ "/PowerService");
} catch { /*tech Error handling*/}
try {
service.doSomething(123) ;
} catch { /* tech and behavioral error handling*/}
[omissis]
```

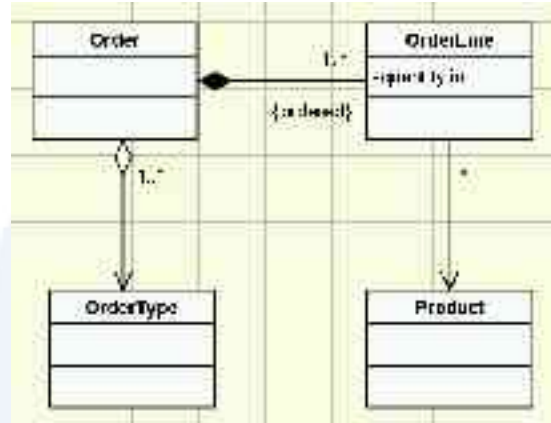
▶ oppure

```
[omissis]
try {
SEE.getService("PowerService").doSomething(123) ;
{ catch { /*behavioral error handling*/ }
[omissis]
```

Il principio di Pareto

- ▶ Vi è un 80% delle funzionalità che deve prendere il 20% del tempo di tempo progetto, poichè avremo un 20% che ne prenderà l'80%!
- ▶ L'80% degli aspetti banali debbono essere affrontati con modelli agilissimi.
- ▶ Esiste però un 20% non può trarre vantaggio perchè in questi casi la complessità non è riducibile a fattori comuni.
- ▶ Ma abbiamo lasciato più tempo per le cose che contano, che danno l'80% dei benefici (e di conseguenza dei danni).
 - ▶ Sono spesso il Core Business del cliente

L'80% triviale



No hand-coding. Flessibilità di cambiare relazioni ed attributi con poco sforzo (CRUDEL).

- ▶ una naming convention sulle chiavi straniere (orderId): *development time*
- ▶ una classe per rappresentare le associazioni (LinkObj): *run time/built time*
- ▶ un repository di metadata: *run time/build time*



Antipattern

- ▶ Architettura funzionale rigida, waterfall per dominio funzionale, architettura tecnica agile
 - ▶ Gli sviluppatori inventano scorciatoie per recuperare
- ▶ Mancanza di un metadata repository
 - ▶ Incapacità di rendere ripetibili i pattern
 - ▶ Incapacità di generare del codice a build time
 - ▶ Incapacità di generare del codice a run-time
- ▶ Rigidità formale, non regolamentare le eccezioni ai principi architetturali
- ▶ Eccedere nel fanatismo idealista dell'astrazione
 - ▶ Es: Wrapping di JMS con APIs proprietarie
 - ▶ OSS: Mozilla+Gnome & Microsoft



Conclusioni

- ▶ Un corpo agile in un'armatura rigida è inutile
- ▶ L'architettura deve consapevole di essere dover essere flessibile
 - ▶ **Assumere che i cambiamenti siano la regola**
- ▶ Un processo agile vuole un'architettura agile
- ▶ Fondamentale non eccedere:
 - ▶ Troppa rigidità
 - ▶ Troppa flessibilità

Grazie

Soluta.net

Via Edificio, 2 Treviso

+39-0423-915547

Pierfranco Ferronato

Chief Architect

pferronato@soluta.net

info@soluta.net



SOLUTA.NET

www.soluta.net

