

A stylized orange figure with a yellow circle above its head and two yellow circles on its arms, set against a white background. The figure is composed of thick, rounded lines.

Architettura del software: andare oltre il design

Lorenzo Barbieri



- Sono un Practice Manager in ObjectWay SpA (www.objectway.it), specializzato in architetture Microsoft .NET, Windows, SQL Server, Visual Studio Team System, Virtual PC/Virtual Server
- Collaboro con UGIdotNET, INETA, Team System Rocks!, e sono tra i soci fondatori di GUISA
 - lorenzo.barbieri@objectway.it
 - www.geniodelmale.info

L'architettura NON BASTA

- Fare un'ottima architettura non basta a garantire che il prodotto finale sia qualitativamente di buon livello.
- Ci sono moltissimi prodotti architeturalmente "ineccepibili" che poi però falliscono miseramente per problematiche che non erano state considerate

TArchitecture e MArchitecture

- I sistemi software hanno DUE dimensioni architettoniche:
 - TArchitecture (Architettura Tecnica)
 - MArchitecture (Architettura Marketing)
- Generalmente con il termine di Architettura Software si intende sempre la TArchitecture

MArchitecture

- MArchitecture (o Marketecture) è la visione orientata al Business dell'architettura di un sistema software.
- Molto spesso questo termine ha un'accezione negativa.
- In questo webcast vedremo le accezioni "positive" del termine, e le sue implicazioni

MArchitecture

- Include:
 - Business Model completo (licenze, modalità di vendita, etc..)
 - Dettagli tecnici rilevanti per i clienti
 - Datasheet
 - Differenziazioni competitive
 - Brand
 - Requisiti di Marketing e Use Case di Marketing

Perchè è importante separarle

- A volte una scelta Architettureale può essere "nascosta" al cliente.
 - Esempio: arch.modulare VS flag di abilitazione
- A volte un vantaggio Architettureale può essere solo "apparente".
 - Esempio: arch.monolitica VS DLL separate
- TArchitecture e MArchitecture DEVONO restare separate!

SICUREZZA



La sicurezza è il primo punto!

- Sempre!
- Sia quando si parla di TArchitecture, sia quando si parla di MArchitecture.
- In questo webcast non parleremo di come progettare un'architettura sicura, non è il tema.
- Ma la sicurezza DEVE essere il PRIMO pensiero degli architetti.

La Security è importante anche per la MArchitecture

- E non solo per le implicazioni tecnologiche.
- Il mercato RICHIEDE la Security (basta guardare gli sforzi fatti da Microsoft per Windows Server 2003, Windows XP Service Pack 2 e Windows Vista)

La Security è a tutti i livelli

- L'impatto sull'architettura della Security è a tutti i livelli.
 - Autenticazioni integrate o con ID Sicuri
 - Standard per lo scambio di dati sicuri, la gestione delle transazioni sicure, l'immagazzinamento sicuro dei dati, etc...
 - Fiducia! Non basta dichiarare di usare gli ultimi strumenti o protocolli sicuri, bisogna creare la fiducia nei propri utenti e clienti.

La Security è la chiave per risolvere le dispute

- L'importanza sempre maggiore dei sistemi software porta all'aumento esponenziale dei possibili problemi e delle possibili dispute legali.
- La Security "attiva" serve per prevenire i problemi
- La Security "passiva" (logging, tracing, sistemi per verificare la non-repudiazione, etc..) è indispensabile in presenza di problemi, anche legali

MODELLI DI BUSINESS



I modelli di Business

- Esistono vari modelli di Business relativi ai progetti software
 - Open Source (e derivati)
 - Far pagare le Release
 - Far pagare l'accesso o l'uso nel tempo
 - Far pagare una percentuale sui ricavi o sui risparmi
 - Far pagare l'uso di servizi collegati

Applicazione e Feature

- In questo contesto intendiamo Applicazione come l'insieme di tutte le Feature del nostro prodotto
- I modelli di Business possono essere applicati a tutta l'Applicazione o solo ad alcune Feature

Bundle o Suite

- Una scelta importante è decidere se creare dei Bundle o delle Suite tra i prodotti che si realizzano:
 - Bundle è semplicemente un insieme scorrelato di prodotti
 - Suite è un insieme correlato e coordinato di prodotti

Suite e MArchitecture

- Decidere se creare un Bundle o una Suite è una decisione che fa parte della MArchitecture, ma il suo impatto è profondo anche per la TArchitecture
- In generale quando si realizzano le Suite c'è un Team che si occupa della MArchitecture di tutta la Suite di concerto con i team che si occupano delle architetture (M/T) dei singoli prodotti

Il Business Model e l'Architettura

- Decidere un Business Model impatta notevolmente sull'Architettura della soluzione
- Progettare un prodotto Open Source, un prodotto commerciale in cui si paga la singola Release o in cui si paga un abbonamento temporale o legato all'uso ha impatti notevoli sui sistemi che si devono realizzare

Un esempio: la scadenza del Software

- Decidere come notificare l'utente che il Software sta scadendo è una decisione che impatta profondamente sul gradimento di un Software da parte di chi lo usa.
- Pensate ad un Software Client/Server dove il Client non scade e invece il Server ha scadenza e da un giorno all'altro non funziona più senza avvisare il client nei giorni/mesi precedenti!

Un esempio: il Pay per Use

- Il Pay per Use è un modello che impatta notevolmente sull'architettura.
 - E' facilissimo da implementare per applicazioni erogate interamente o parzialmente in ASP (Application Service Provider)
 - E' più difficile da implementare per applicazioni completamente indipendenti
 - Bisogna creare dipendenze con l'esterno per il tracking dell'utilizzo
 - Bisogna evitare disservizi in caso di mancanza di collegamento o altri problemi tecnici

SERVICE LEVEL AGREEMENT



Service Level Agreement

- **Service Level Agreement (SLA)** è la parte di un contratto di erogazione di un servizio che definisce il livello che il servizio deve avere.
- Generalmente si riferisce a campi come:
 - Performance, Affidabilità, Carico, Etc...
- Il SLA definisce la maggior parte dei requisiti non funzionali o di Quality of Service

TIPOLOGIE DI LICENZE



Le licenze impattano l'architettura?

- Sì!
- C'è la licenza del Nostro software, la licenza dei componenti che utilizziamo, la licenza (o le licenze) dei servizi che decidiamo di utilizzare, etc...
- Decidere il tipo di licenza del Nostro software porta a fare scelte architettoniche potenzialmente diverse

Un esempio: componenti open source

- Un classico esempio riguarda l'utilizzo di componenti Open Source.
- Esistono varie licenze Open Source:
 - Alcune, come le licenze di tipo BSD, quelle proposte da Microsoft e altre, non impongono particolari restrizioni al software che si sta realizzando
 - Altre, come le licenze "virali" richiedono che tutto il prodotto che usa il componente Open Source venga rilasciato come Open Source

PORTABILITÀ



Portabilità

- La Portabilità è un punto in cui di scontro tra MArchitecture e TArchitecture.
- Di solito il marketing vorrebbe un prodotto “portabile” senza comprendere effettivamente il vero significato della portabilità.

Portabilità del Front-End

- E' la più difficile da realizzare
 - Microsoft .NET -> Mono, non è ancora completo
 - Java, interfaccia utente molto diversa da quella nativa
 - C++ etc..., grande fatica per adattarsi alle varie piattaforme

Portabilità della Business Logic

- Molto spesso non conviene rendere la Business Logic portabile
- Conviene usare interfacce standard verso il Front-End (ad esempio Web Services) e utilizzare uno o più server nella tecnologia scelta
- Costa MOLTO di MENO, ed è generalmente accettato.

Portabilità dell'accesso ai dati

- Molto spesso è richiesto di integrare il sistema con il Database esistente.
- In questo caso il costo di adeguamento è giustificato dal dover mantenere un'unica piattaforma a livello aziendale e da non dover acquisire altre licenze.

Decidere se ne vale la pena

- Si può usare la cosiddetta "Matrix of Pain":

	Windows 2000 Server	Windows Server 2003	HP-UX	AIX
SQL Server 2000	X	X		
SQL Server 2005		X		
Oracle 8i	X			X
Oracle 9i		X	X	

Attenzione alle promesse che si fanno!!!

- Portabilità è una parola che riempie la bocca, ma che può portare ritardi e notevoli aumenti di costi
- Attenzione che supportare piattaforme obsolete (Windows 9x, vecchie versioni di Unix, Mac OS < X, etc...) può essere veramente impegnativo e non è detto che si ottenga più il supporto dalle case produttrici in caso di problemi

ESTENSIBILITÀ ED INTEGRAZIONE



Estensibilità ed Integrazione

- Estensibilità ed Integrazione fanno la differenza tra un prodotto ottimo ma poco utilizzabile nell'ambiente esistente e un prodotto buono ma integrabile con l'ambiente
- E' una delle richieste più ricorrenti dei clienti, ed una delle "Feature" che più vengono considerate dai Decision Maker

Impatti sull'Architettura

- Gli impatti sull'Architettura sono notevoli:
 - Service Oriented Architectures
 - Application Blocks
 - Aspect Oriented Programming, Inversion of Control, etc...
 - Modello a Plug-In
 - Enterprise Service Bus
 - Workflow Foundation, BPEL4WS, Etc...
- Sono tutte tecnologie per permettere l'integrazione e l'estensibilità

Impatti sull'architettura futura

- La scelta di aprire delle API, aggiungere punti di estensibilità, estrarre la logica in flussi esterni, etc..., è una scelta coraggiosa, ma introduce delle limitazioni nelle evoluzioni future che bisogna considerare attentamente

BRAND



Brand

- Ogni prodotto ha un suo Brand
- La regola d'oro quando si deve scegliere il nome del prodotto e il brand è:

Gli sviluppatori devono scegliere i nomi delle classi, non i nomi dei prodotti!

Nomi in codice

- Diverso è il discorso dei nomi in codice dei prodotti quando sono ancora in fase di sviluppo.
- Di solito il target è diverso, si parla di Early Adopters, che sono persone molto tecniche.

**Gli sviluppatori possono scegliere i
Codename!**

Il Brand non è solo il nome

- Interfaccia grafica, usabilità, semplicità, etc...
- Tutto questo concorre al Brand del prodotto.
- E' per questo che all'interno dei Team servono esperti nella User Experience, per poter gestire efficacemente questi aspetti.

INSTALLAZIONE E GESTIONE DEGLI AGGIORNAMENTI



Installazione

- Molto spesso l'installer è l'ultimo componente che viene sviluppato prima di rilasciare un prodotto, e molto spesso non viene "progettato" ma semplicemente "realizzato"

L'Installazione è una fase cruciale

- La cosiddetta "Out Of the Box Experience" (OOBE) è una fase cruciale nella vita di ogni prodotto, in particolare dei prodotti software, e ancor di più di quelli che vengono distribuiti in forma di versioni di prova o di demo

Aggiornamento

- Per capire come gli utenti vedono l'aggiornamento vale questa regola:

L'aggiornamento è simile all'installazione, solo 100 volte peggio!!!

Aggiornamenti

- Non basta “progettare” l’installazione, bisogna anche “progettare” il sistema per gestire gli aggiornamenti.
- Anche in questo caso l’impatto sull’architettura è presente, bisogna “progettare” il sistema per poter usare tecnologie come ClickOnce o l’Updater Application Block.

Aggiornamenti delle componenti server

- L'aggiornamento delle componenti server richiede ancora più attenzione, perchè errori in queste fasi potrebbero portare a perdite di dati o altre conseguenze molto dannose
- Bisogna adottare sistemi per la "gestione" dei rilasci, anche e soprattutto per i componenti server e i dati

Aggiornamento ad una versione più potente

- Oltre all'aggiornamento tra versioni diverse (1.x -> 2.0) bisogna rendere semplice il passaggio tra due edizioni diverse della stessa versione (Standard -> Enterprise ad esempio)

CONFIGURAZIONE E VERSIONI



Configurazione e Versioni

- La gestione della Configurazione dei software, soprattutto in caso di versioni diverse è molto critica e va progettata bene.
- Nessuno vuole dover reinserire tutta la Configurazione solo perchè ha acquistato una versione più potente.

Configurazione

- La Configurazione DEVE poter essere salvata e reimportata, soprattutto nel caso di applicazioni Enterprise con decine o centinaia di client installati.
- La Configurazione DEVE essere sicura, per evitare compromissioni di dati sensibili.

GESTIONE DEI RILASCI



Rilasci e RoadMap

- Un punto fondamentale della MArchitettura che influenza profondamente anche la TArchitettura è la gestione dei rilasci e la RoadMap dei rilasci successivi.
- I clienti devono avere ben chiaro in che direzione sta andando il prodotto, e lo stesso vale per gli architetti e i progettisti, che non devono limitarsi a considerare la versione attuale.

Tempi

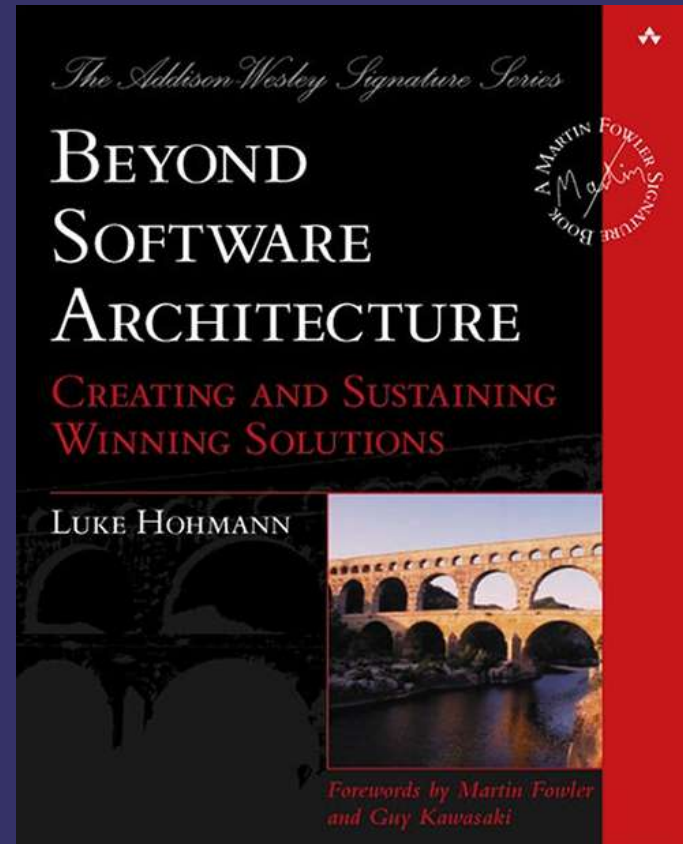
- Molto spesso nella MArchitecture si tende a parlare al *passato* anche quando si parla della release corrente, perchè si è già focalizzati sulle release successive.
- Questo a volte è un male, perchè si dà l'impressione di non supportare a dovere la release corrente. Conviene parlarne sempre al *presente*.
- A volte, in contesti circoscritti, parlarne al passato è utile per far vedere l'evoluzione.

PER SAPERNE DI PIÙ



Libro consigliato

- Questo webcast è liberamente ispirato dal libro **Beyond Software Architecture – Addison Wesley**.
- Il libro ha un taglio diverso, ma tratta gli stessi argomenti trattati nel webcast



Visual Studio 2005 è disponibile: scegli il prodotto più giusto per te
www.microsoft.it/msdn/vs2005/

- **Visual Studio 2005 Team Edition**

- Visual Studio Team Edition (for Architects, Developers o Testers) con MSDN Premium
- Visual Studio Team Suite con MSDN Premium
- Visual Studio Team Foundation Server

- **Strumenti professionali**

- Visual Studio 2005 Professional
- Visual Studio 2005 Professional con MSDN Professional
- Visual Studio 2005 Professional con MSDN Premium
- Visual Studio 2005 Tools for Microsoft Office System

- **Strumenti di base**

- Visual Studio 2005 Standard
- Visual Studio 2005 Express Edition

- **Altri strumenti**

- Visual SourceSafe 2005
- VisualFox Pro 9.0

**Licenze individuali:
1 sviluppatore = 1 licenza**

Dove acquistare:

www.microsoft.it/msdn/rivenditori/

Per informazioni:

itamsdn@microsoft.com

Ricordatevi di compilare il modulo di Feedback

DOMANDE?

