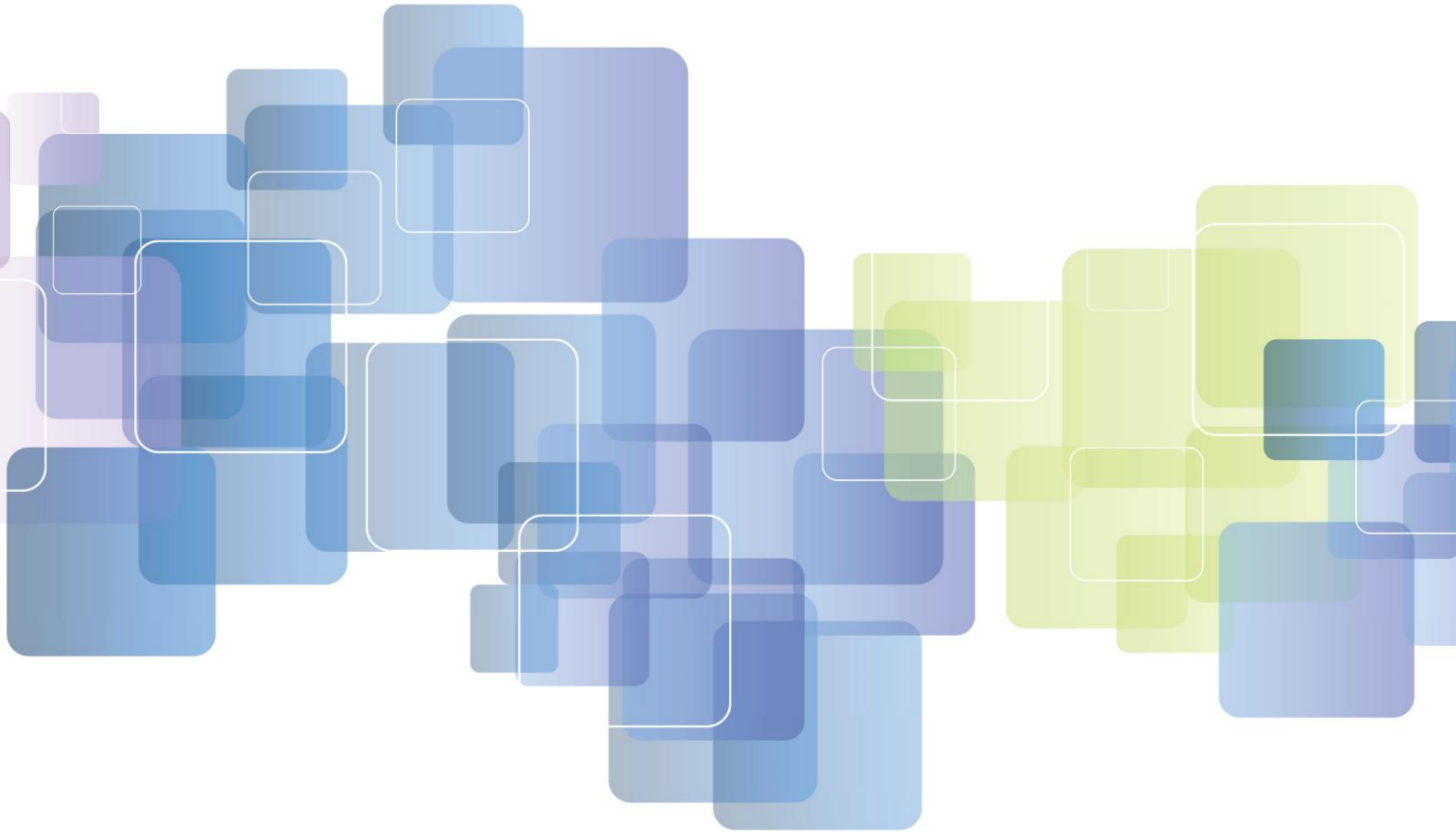
TecnetDati

La riusabilità questa sconosciuta

Emilio C. Porcelli



E' la grande promessa dell'OO, non datela per scontata!

Gli scozzesi - si sa - hanno fama di essere gente avveduta. E' infatti il primo a parlare di riuso sistematico del software è stato Dough McIlroy, nome dal suono inequivocabilmente scozzese. Era il 1968.
Nell'ottobre di quell'anno la NATO riunì infatti a Garmisch, sulle Alpi Bavaresi, un gruppo di esperti per dibattere di quella che già allora veniva chiamata la "crisi del Software", e dei modi per risolverla. Si era ancora in piena guerra fredda e la difesa del Mondo Libero passava anche di lì. E' in quell'occasione che vennero coniati termini come Ingegneria del Software e Software Factory. E sempre in quell'occasione McIlroy, allora alla Bell Telephone, preconizzò una produzione di massa del software basata su componenti: "Gli ingegneri del software dovrebbero prendere spunto dall'industria dell'hardware per sviluppare una sottoindustria delle componenti software. Queste componenti non dovrebbero essere prodotte in base alle esigenze del momento, ma per poter essere riusate in una pluralità di sistemi". Da allora sono trascorsi trent'anni, ma in tutto questo tempo la riusabilità del software ha fatto pochi incerti passi in avanti. Tanto da indurre qualcuno a commentare: "E' come se la prima tappa della strada che conduce a un riuso sistematico del software fosse stata percorsa trenta volte". Chiediamoci allora perché. I perché sono tanti, di natura tecnica e non. Anzi i maggiori inibitori della riusabilità non sarebbero di natura tecnica, ma piuttosto organizzativa, culturale e perfino psicologica - la sindrome del "Not Invented Here" insegna. Ma anche i fattori tecnici hanno il loro peso. A provarci sono stati in tanti, ma vuoi per le limitazioni dei linguaggi tradizionali, vuoi per la oscurità delle interfacce, le librerie di moduli riusabili raramente hanno dato buona prova di sé.
E' tutto vero, anzi verissimo - dirà qualcuno -, ma oggi le cose sono finalmente cambiate. Infatti con l'Orientamento agli Oggetti e grazie a ereditarietà, overloading e polimorfismo gli ostacoli di natura tecnica, almeno quelli, sono stati definitivamente rimossi e così la riusabilità non è più un miraggio. Tanto che Adele Goldberg e Kenneth Rubin, due personaggi che di queste cose se ne intendono, citando in Succeeding with Objects: Decision Frameworks for Project Management (Addison-Wesley, 1995) i risultati di una loro indagine su un campione di 23 progetti software non banali condotti da 19 aziende statunitensi, segnalano la riusabilità come uno dei fattori - anche se non il più importante - che hanno indotto queste aziende ad abbandonare gli sviluppi tradizionali per passare agli Oggetti. Con quali risultati? Spesso eccellenti, ma non sempre sotto il profilo della riusabilità.

Il problema della gestione dei titoli di coda

Avete idea, vedendo sfilare i titoli di coda di una qualsiasi produzione televisiva, quanti e quali siano i contratti in gioco? Non si tratta solo di compensare registi, attori e comparse; ci sono anche i truccatori, i parrucchieri e tutta quella variegata schiera di personaggi che affollano un set televisivo. Per non parlare del noleggio degli abiti e degli arredi di scena. Raccapezzarsi tra tutti questi contratti non è un problema da poco, ma è in casi del genere che l'Orientamento agli Oggetti dà il meglio di sé. E infatti c'é chi, qui in Italia, ha pensato bene di risolverlo commissionando all'esterno una bella applicazione Smalltalk. Con quali risultati? Il software - potenza degli Oggetti! - è stato realizzato nei tempi e ai costi previsti e, ciò che più conta, con piena soddisfazione del committente e degli utenti. Soddisfazione sì, ma solo fino a un certo punto. A cose fatte si è infatti scoperto che le classi sviluppate erano del tutto inutilizzabili in un qualsiasi contesto che non fosse esattamente la gestione dei contratti di quelle produzioni televisive.

La risposta dell'Agenzia Spaziale Europea

Fatti del genere rappresentano la regola più che l'eccezione. John Favaro dell'Agenzia Spaziale Europea (ESA) ha infatti raccolto un'ampia casistica di progetti OO commissionati all'esterno i cui artefatti si sono poi rivelati tutto fuor che riusabili. Se la proprietà del software resta del committente - osserva Favaro -, chi lo sviluppa sarà ben poco incentivato a renderlo riusabile. La soluzione? Quella adottata da ESA è molto semplice: il software resta di proprietà di chi lo ha prodotto e il committente ne acquista una licenza d'uso non esclusiva. Sarà, ma mi piacerebbe vedere la faccia di certi produttori di software, per non parlare dei loro clienti, di fronte alla prospettiva di uno strappo così radicale rispetto ai loro modi di porsi rispetto al mercato.

La riusabilità costa. E costa di più

Il punto è che sviluppare una componente software riusabile costa e costa di più. Per ingegnerizzarla, generalizzarla, testarla e documentarla occorre più tempo, fatica e denaro di quanto non richieda lo sviluppo di una "normale" componente. Quanto di più? Will Tracz in base alla sua esperienza di venditore di software "usato" (Proprio così! Vedi Confessions of a Used Program Salesman, Addison-Wesley, 1995) parla di aggravi di costo del 60%, con punte del 200% registrate da Lockeed Martin e IBM per alcuni loro progetti. Ecco perché, una casa di software sarà ben poco incentivata a sostenere il maggiore sforzo richiesto per rendere riusabile il software che produce.
Ma il problema non riguarda solo il software commissionato all'esterno, si ripropone infatti puntualmente anche per gli sviluppi condotti internamente alla Aziende. Mettiamoci nel panni di un Capo progetto. La sua è una posizione ben poco invidiabile: i tempi di consegna sono normalmente stringenti e le risorse su cui può fare conto spesso ridotte all'osso. Chiedergli quello sforzo in più che la riusabilità comporta sarebbe pretendere troppo. Sviluppi in ottica progettuale e riusabilità sembrerebbero quindi difficilmente conciliabili. E' un vero peccato perché se è vero che la riusabilità è un costo, è anche vero che produce consistenti risparmi. Si stima infatti che cinque linee di codice riusato costino mediamente quanto una linea di codice sviluppata ex novo. Sempre secondo Tracz, posto uguale a 1 il costo dello sviluppo di una nuova componente, il Costo Relativo di Riuso (CRR) - e cioè il costo del riuso di una

componente similare già esistente senza che la si debba modificare (è il cosiddetto riuso a "scatola nera"), sarebbe compreso tra 0,1 e 0,4. Detto per inciso, un CRR di 0,4 sarebbe caratteristico di componenti come i menù e le maschere video il cui inserimento in una nuova applicazione risulta più oneroso. Tutto ciò naturalmente a patto che la componente non debba essere modificata. Il risparmio offerto da un riuso a "scatola nera", che è mediamente del 20-25%, crolla infatti al 5% per un riuso a "scatola bianca" (la componente va modificata) e se le modifiche vanno al di là di un certo limite può trasformarsi in una perdita secca.
E allora? Se volete percorrere senza inconvenienti la strada della riusabilità, tenetevi ben a mente le Tre Regole d'Oro che troverete nel riquadro su questa stessa pagina.

LE TRE REGOLE DEL TRE

Prima Regola - Per poter riusare qualcosa, dev'esserci qualcosa da riusare.

Seconda Regola - Prima di poter sviluppare un software riusabile, occorre svilupparlo tre volte.

Terza Regola - Prima di poter trarre beneficio dal suo riuso, occorre riusarlo almeno tre volte.

Tecnet Dati s.r.l.
C.so Svizzera 185 –
10149 - Torino (TO), Italia
Tel.: +39 011 7718090 Fax.: +39 011 7718092
P.I. 05793500017 C.F. 09205650154
www.tecnetdati.com