ACQUISIRE E SVILUPPARE SOFTWARE MILITARE CON COMPONENTI *OPEN SOURCE*

Indagine su processi iterativi, incrementali, agili e per COTS *open source* applicati allo sviluppo e all'acquisizione di sistemi software per la Pubblica Amministrazione e le Forze Armate italiane

di Salvatore Alessandro Sarcià

L'introduzione di sistemi informatici (C2, contabili, stipendiali, finanziari e di controllo) nella Pubblica Amministrazione e nelle Forze Armate rappresenta oggi la chiave di volta dell'efficienza, efficacia ed economicità delle Amministrazione dello Stato. Tuttavia, sviluppare/acquisire sistemi SW è una attività particolarmente complessa che, spesso, può sfociare in insuccessi molto costosi (il SW non fa quello che si voleva inizialmente, c'è un significativo aumento dei costi, le scadenze non sono rispettate e/o la qualità richiesta non è garantita). In questo articolo vengono in primo luogo analizzate le ragioni più comuni di questi insuccessi e successivamente approfondite le possibili soluzioni per evitare che tali insuccessi possano verificarsi. Lo studio proposto è calato sulla specifica realtà della Pubblica Amministrazione e delle Forze Armate italiane con un occhio a quanto avviene in ambito internazionale. Particolare enfasi è stata posta nell'illustrare le metodologie di sviluppo agile e basate sul riuso di componenti open source. Da ultimo, l'articolo illustra gli elementi di dettaglio su come integrare, in pratica, i contratti di acquisizione/sviluppo pubblici in senso iterativo, incrementale ed agile.

Lo sviluppo del software (SW) è senza dubbio una delle più affascinanti quanto complesse attività che l'uomo abbia mai intrapreso nella sua plurimillenaria storia. Ancora oggi, nelle comunità scientifiche[1] esiste, tra i tanti aspetti insoluti, una famosa controversia riguardante il fatto se la creazione di SW sia da considerarsi scienza o, come la parola stessa ("creazione") suggerirebbe, arte. I fautori del SW come scienza preferirebbero certamente usare il termine "produzione" poiché evocherebbe molto di più gli aspetti riferiti alla misurabilità e riproducibilità del processo di sviluppo utilizzato e del prodotto realizzato.

Ciò che necessariamente preme considerare è come si debba trattare lo sviluppo del SW a livello organizzativo ed industriale. Se si considera lo sviluppo ("creazione") di SW come un'arte, allora, parleremmo di "artista" e non d'ingegnere che sarebbe libero di creare la sua "opera" (d'arte) in base ai suoi personali gusti, seguendo le sue categorie e peraltro nessuno potrebbe sindacare la bontà delle sue scelte. La creazione sarebbe allora

[1] Per approfondimenti, si consultino le biblioteche digitali IEEE, ACM ed Elsevier.

Software Gestionali liberi

un fatto personale, non un processo di team necessariamente ripetibile né tanto meno misurabile rispetto a standard qualitativi di riferimento (quanto è il tempo richiesto per realizzare un'opera d'arte?). Se invece decidessimo di trattare lo sviluppo del SW come un fatto puramente ingegneristico potremmo usare categorie scientifiche anche se non sempre e non completamente applicabili a tutti gli aspetti del processo produttivo. È su questi concetti che si è sviluppata negli ultimi decenni una delle più promettenti ed innovative discipline dell'Informatica che, non a caso, è stata chiamata Ingegneria del Software (*Software Engineering*). Oggi, il termine ingegneria del SW sta ad indicare quell'insieme di processi, pratiche e procedure nell'analisi, progettazione, sviluppo e mantenimento del SW e lo studio di tali discipline. Il termine fu creato nell'ambito di Conferenze della NATO[2] (Garmisch, Germania, 7-11 ott. 1968 e Roma, Italia, 27-31 ott. 1969) con lo scopo di far riflettere la comunità scientifica circa la c.d. "crisi del SW". Infatti, fin da allora si capì che sviluppare SW (militare, aero-spaziale, scientifico, industriale, bancario etc.) di qualità fosse un'attività che avrebbe dovuto seguire un approccio puramente ingegneristico con regole, pratiche e paradigmi ben fondati come quelli usati, appunto, nell'ingegneria classica (infrastrutture, meccanica ed elettronica). Infatti, i grandi e costosi fallimenti nello sviluppare SW affidabile, "privo" di difetti, nei tempi stabiliti e con le funzionalità concordate si rilevò ben presto una impresa molto complessa se non addirittura impossibile. L'approccio empirico e sperimentale[3] all'ingegneria del SW rispetto a quello classico prevede che prima di ritenere una tecnica valida, essa debba essere testata attraverso una serie di sperimentazioni, cioè attraverso protocolli sperimentali identici a quelli usati nell'ambito della medicina per verificare, at-

[2] Sponsorizzata dal NATO Science Committee.
[3] Il Goal-Question-Metrics (GQM) è uno dei principali pilastri su cui si fonda tale approccio.

traverso test di confidenza, l'efficacia di un trattamento rispetto ad altri.
Tra tutte le tecniche esistenti, l'articolo si focalizzerà specificatamente su quattro di esse e cioè: 1) *approccio iterativo* rispetto a quello *a cascata*, 2) *incrementale* (con incrementi intermedi funzionanti) rispetto a quello della *soluzione unica finale*, 3) *agile* rispetto a quello *istituzionalizzato* ma più pesante da eseguire, 4) basato su *COTS*[4] *open source*[5] rispetto a quello *senza componenti* o con *componenti proprietari*.

Software Gestionali

La domanda cui s'intende rispondere è quindi se, in base alle attuali norme (italiane/europee) sui contratti, sia legalmente ammissibile sviluppare/acquisire sistemi SW per la Pubblica Amministrazione attraverso metodologie iterative, incrementali e agili basate su COTS *open source*.

PROCESSO DI SVILUPPO AGILE

La scelta del processo di sviluppo è una scelta critica per il successo del progetto di sviluppo/acquisizione del SW. Il processo scelto è il principale elemento per implementare una metodologia agile. L'elevata complessità, la limitatezza del tempo e delle risorse a disposizione, come detto sopra, rendono lo sviluppo del SW assai rischioso. Esistono molti studi, tra cui "Chaos Report 2009" dello Standish Group, che mostrano che il 68% dei progetti SW non rispetta i criteri per cui era stato concepito in termini di scadenze, budget e funzionalità/qualità. In particolare, il 33,8% di questo 68% viene addirittura cancellato prima che sia terminato. Solo il 32% dei progetti ha realmente successo in termini di rispetto dei criteri detti sopra. L'elevato tasso di fallimento è dovuto a molti fattori legati alla man-

---

[4] COTS sta per "Component Off The Shelf", cioè componenti SW già pronti ed usabili direttamente per integrazione/collegamento con il resto del sistema sviluppato.

[5] Non si intende suggerire lo sviluppo di SW open source all'interno delle istituzioni nazionali, ma il riutilizzo di componenti, già sviluppati e testati dalle comunità open source, nell'ambito di sistemi SW che devono essere sviluppati per lo Stato.

Software oper source *(Copyright Neosidea)*

cata comprensione del problema, all'inadeguatezza del team di sviluppo, all'intrinseca complessità, alle stime sui tempi di consegna completamente irrealizzabili rispetto al budget, alla scarsa comunicazione tra sviluppatori e committenza e alla mancanza di input da parte del cliente solo per citarne alcuni. Il fattore costante in tutti i progetti che falliscono o che comunque non vengono consegnati secondo quanto pattuito è data dall'incompletezza dei requisiti.

Immagine iconografica delle attività Gestionali *(Copyright Trizero)*

Per contro, sempre sulla base del "Chaos Report 2009", sono stati evidenziati i punti di forza su cui i progetti di successo normalmente si basano, che sono:

1 coinvolgimento degli utenti;
2 supporto da parte dell'alta dirigenza;
3 requisiti espressi con chiarezza;
4 adeguata pianificazione;
5 scadenze realistiche;
6 project Milestones più brevi;
7 competenza del team di sviluppo;
8 l'essere proprietari del sistema da sviluppare;
9 obiettivi e visione chiari;
10 lavoro intenso e focalizzato del team di sviluppo.

La metodologia del processo a cascata (Figura 1), secondo la quale prima si effettua in sequenza analisi dei requisiti, design, implementazione, integrazione e test di sistema non è la migliore scelta per lo sviluppo dei sistemi normalmente sviluppati/acquisiti dallo Stato.

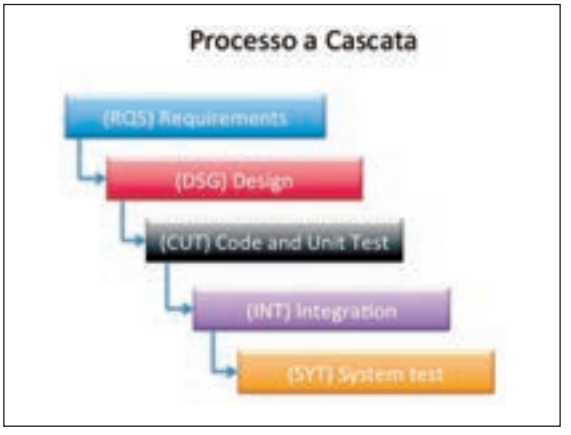

Figura 1. Processo a cascata.

O meglio, è probabile che per sistemi di piccole dimensioni per cui gli sviluppatori siano particolarmente esperti e perfettamente a conoscenza del dominio applicativo (due fattori difficilmente ottenibili simultaneamente), il processo a cascata potrebbe anche essere utilizzato, ma per i sistemi SW di dimensioni e/o complessità elevati, la scelta ricade senza dubbio su quelli iterativi, incrementali (Figura 2) ed agili basati intensivamente su continue integrazioni, test e rilasci.

Figura 2. Processo iterativo con nomi di attività mutuate dal processo a cascata.

Il punto essenziale è dunque quello di comprendere che nessuno è in grado di conoscere a priori quali requisiti SW debbano/possano essere effettivamente sviluppati. Come si può allora scrivere un allegato tecnico esaustivo da inserire nel contratto? Prima di tutto va capito se sia conveniente mettere a punto un contratto nel quale sia necessaria la presenza di un allegato tecnico dettagliato con tutti i requisiti SW da sviluppare. Ciò che si vuole mettere in evidenza è che, cercare di fissare (a priori) in un contratto i requisiti di dettaglio del sistema è forse la cosa più deleteria che si possa fare ai fini del successo del progetto. La chiarezza contrattuale deve essere riposta nel descrivere i bisogni utente (*user needs*), cioè descrivere dettagliatamente, analiticamente e in maniera non ambigua i problemi che si intendono risolvere.

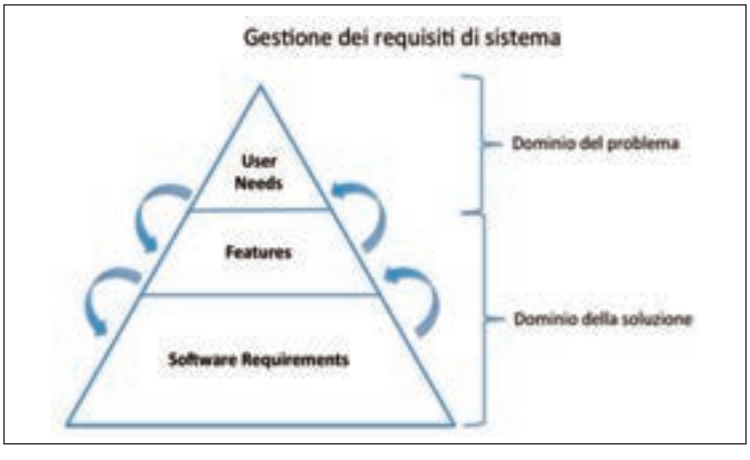

Figura 3. Schematizzazione del processo di gestione dei requisiti di sistema (Leffingwell D. and Widrig D., 2003).

Si noti infatti (Figura 3) che la definizione delle funzionalità del sistema (*features*) e i requisiti del software (*software requirements*) appartengono al dominio della soluzione e quindi sono di **competenza degli sviluppatori e non della committenza**. Nel processo iterativo l'accettazione dei requisiti da sviluppare avviene a processo iniziato e non a priori sul contratto. Preso atto di ciò, allora, la scelta che limita il rischio di fallimento è quella di rimandare la decisione su cosa effettivamente sviluppare (definizione dei requisiti software) a quando si realizzerà l'analisi del dominio del problema in maniera approfondita e si sarà passati quindi a sviluppare il sistema. Allora l'accordo contrattuale iniziale potrebbe essere effettuato sul "perché" (problema da risolvere espresso in maniera chiara ed inequivocabile), sul "quanto" (numero di funzioni da sviluppare in un certo tempo), sulla qualità (conformità a standard e requisiti non funzionali) e sul "come" (processo di sviluppo), ma solo parzialmente sul "cosa" (descrizione dei soli *user needs* e non dei requisiti SW). Ovviamente, una parte dell'accordo a priori dovrebbe essere anche quella di definire le modalità di coordinamento all'interno della singola interazione tra committenza/analisti di sistema e sviluppatori (un approccio agile molto in voga oggi è SCRUM, ma di approcci agili ne esistono una varietà di "dialetti"). Il contratto dovrà quindi prevedere la negoziazione/accettazione dei requisiti SW in corso d'opera. Si noti tuttavia che accanto alle funzionalità si dovrebbe comunque tener conto dei requisiti non funzionali e di qualità. Ad ogni termine di fase si conseguono delle *milestone* ed il sistema, in maniera appunto incrementale, cresce sempre più fino ad avvicinarsi al suo stato finale del 100% (Figura 4).

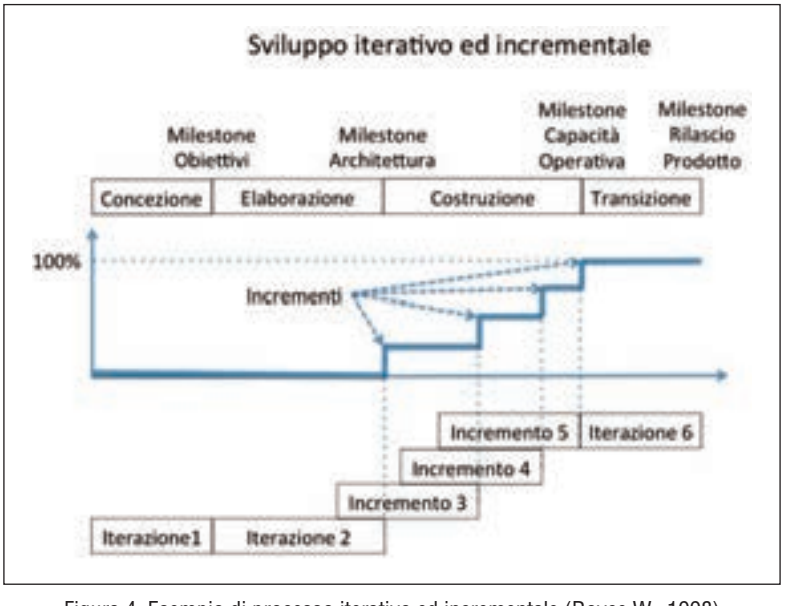

Figura 4. Esempio di processo iterativo ed incrementale (Royce W., 1998).

L'attore agisce esclusivamente sulla interfaccia (*view*), mentre il sistema, intercettando le azioni dell'attore eseguite sull'interfaccia attraverso un controllore di interfaccia (*controller*), reperisce i dati (*model*) memorizzati nel sistema o forniti da altri attori/sistemi applicandovi delle azioni per dare risposta, attraverso l'interfaccia, alle azioni invocate dall'attore principale del caso d'uso. Questo schema architetturale (detto *pattern*) che garantisce la robustezza di una architettura SW è detto appunto *Model-View-Controller* (MVC) ed è stato sviluppato per il SW orientato agli oggetti.
Sia che si utilizzino metodologie classiche (*use cases*) che puramente agili (*user's stories*) l'aspetto rilevante è che il sistema sviluppato sia in grado di coprire sempre più gli *user needs* dai quali gli sviluppatori avevano mutuato i requisiti SW come raffigurato in Figura 3.

PRINCIPI DEL PARADIGMA "AGILE" DA INCLUDERE NEI CONTRATTI PUBBLICI

Ciò che si intende mostrare è che i principi del paradigma Agile illustrati di seguito sono completamente compatibili con il codice dei contratti italiano/europeo e che quindi non esiste alcuna buona ragione per non seguire tali principi recependoli all'interno dei contratti della PA/FA. I principi sono i seguenti:

1 *La principale priorità è di soddisfare i clienti attraverso la distribuzione di SW di valore in maniera puntuale e continua.* Per soddisfare il cliente il punto centrale è quello di fornire SW continuamente e senza ritardi e che sia soprattutto di valore per il cliente.
2 *Il cambiamento dei requisiti è benvenuto, anche se avviene in ritardo. I processi agili usano il cambiamento per sostenere il vantaggio competitivo dei clienti.* L'aspetto del cambiamento dei requisiti è centrale. Infatti, il paradigma Agile presenta il cambiamento non come "un nemico da combattere" ma la costatazione del fatto che nessuno, neanche il cliente esperto del business è in grado di comprendere subito e completamente cosa debba fare il sistema in via di sviluppo. È evidente tuttavia che essere in grado di gestire il cambiamento usandolo come un'occasione di miglioramento piuttosto che come una situazione di crisi da gestire non è semplice e richiede alta professionalità.
3 *Distribuire SW funzionante frequentemente, da un paio di settimane a un paio di mesi, cercando di ridurre questo intervallo temporale.* Si è in pratica compreso che affinché un progetto di sviluppo/acquisizione di SW abbia successo, gli sviluppatori devono distribuire (incrementalmente) SW effettivamente funzionante, ancorché incompleto, con un ritmo stabilito che va appunto dalle due settimane (alcuni sostengono anche una settimana) fino ad un massimo di due mesi, alcuni sostengono massimo 1 mese.
4 *Gli analisti del business (esperti) e gli sviluppatori devono lavorare insieme giornalmente lungo tutto il progetto di sviluppo/acquisizione del SW.* Il continuo scambio di informazioni tra gli esperti di business e sviluppatori è una condizione necessaria per avere successo (feedback continuo).

Software oper source

5 *I sistemi vanno costruiti e sostenuti all'interno dell'organizzazione da persone motivate ad avere successo.* Tuttavia, a queste persone vanno dati strumenti, risorse, ambiente e fiducia di cui necessitano per avere successo nel senso che la motivazione non basta.
6 *La conversazione faccia a faccia è il modo (scientificamente) più efficiente ed efficace per far arrivare e circolare l'informazione al team di sviluppo.* Ciò non significa che altri metodi non siano efficaci (usare sistemi elettronici di comunicazione come email, documenti, *conference call* etc.).
7 *Il paradigma Agile prevede che lo sviluppo sia sostenibile anche nel lungo periodo nel senso che sponsor, sviluppatori e utenti dovrebbero essere in grado di mantenere un passo costante nello sviluppo indefinitamente.*
8 *L'attenzione continua all'eccellenza tecnica e alla progettazione rendono più efficace il paradigma Agile come pure la semplicità (intesa come l'arte di massimizzare la quantità di lavoro ancora da svolgere) è ritenuta essenziale.*
9 *La ricerca ha appurato che i migliori risultati in termini architetturali, dei requisiti e di progetto vengono da team di sviluppo che hanno saputo auto-organizzarsi in base alle esigenze e tenendo a mente i principi dello sviluppo Agile.*
10 *Il team di sviluppo Agile è tale se diventa una sorta di organizzazione che apprende nel tempo (learning organization) poiché a intervalli regolari deve riflettere su come diventare più efficace aggiustando e calibrando i propri comportamenti.*

Per valutare il grado di *compliance* di un certo processo di sviluppo rispetto al paradigma Agile esistono delle dimensioni fornite da McConnell che sono:

- **Rapporto % tra personale Junior e Senior**: 0%-35% (cioè 0% Junior e 35% Senior) è il miglior rapporto per incontrare i principi Agili, mentre 40%-15% è la peggior situazione, una intermedia è 20%-25%.
- **Cambiamento dei requisiti**: è una scala da 50 (migliore) a 0 (peggiore, dove i requisiti non cambiano mai), una intermedia è 10.

Software per Ufficio Oper Source

- **Cultura dell'organizzazione che effettua lo sviluppo (non riferito alla committenza)**: la migliore situazione è quella in cui gli sviluppatori lavorano per divertimento, la peggiore è quella in cui ci sono conflitti e caos.
- **Dispersione del team di sviluppo**: più è alta in termini di tempo, distanza fisica e culturale e peggio è.
- **Dimensione numerica del team di sviluppo**: la scala va da 3 sviluppatori (migliore) a 300 (peggiore).
- **Impatto del prodotto (perdite dovute ai difetti):** la scala parte dalla migliore situazione per il paradigma Agile (confort) in cui a causa dei difetti non ci sono perdite, fino alla peggiore in cui ci sono perdite di vite umane.

In termini operativi le principali pratiche in linea con i principi riportati nel manifesto da prendere a base per scrivere un contratto di sviluppo di SW con tecniche Agili sono le seguenti:

- la PA/FA e la ditta fornitrice concordano di sviluppare il SW secondo un processo iterativo ed incrementale (come descritto sopra) in cui, quindi, i requisiti SW cambieranno durante lo sviluppo senza aggravi di costi né per la PA/FA né per la ditta implicando che nessuno può conoscere a priori i dettagli delle funzionalità da sviluppare;
- la ditta fornitrice deve rilasciare alla PA/FA incrementi di SW funzionante a intervalli fissi e ridotti (ad es. 1-4 settimane);

- la PA/FA deve ricevere una pianificazione scritta preventiva di tutto il progetto sia a lungo (2-12 mesi) che a breve termine (30-60 giorni) consentendo anche una certa flessibilità;
- lo sviluppo fatto dalla ditta deve prevedere cicli di sviluppo fissi (time-boxes chiamati sprints o iterazioni), ad esempio 30 giorni, da rendere compatibili con i rilasci di SW funzionante di cui sopra;
- i team di sviluppo devono includere tutte le professionalità richieste (cross-funzionali) inclusi gli esperti di dominio normalmente appartenenti alla PA/FA;
- la PA/FA deve espressamente richiedere alla ditta sviluppatrice la presenza a tempo pieno di un coach (esperto del paradigma Agile) che guidi il team (active management), in maniera "non invasiva", nell'applicazione delle tecniche agili;
- la PA/FA deve richiedere l'adozione di standard per il codice (le best practice del software engineering reperibili in qualsiasi manuale ivi inclusa la stima dell'effort richiesto);
- la PA/FA deve specificare che la ditta faccia continuamente (anche su base giornaliera) l'integrazione di sistema e i relativi test, la ditta deve essere in grado di esibire i risultati dei test eseguiti;
- è altresì necessario che la ditta effettui test di regressione unitari con appositi strumenti di automazione;
- la PA/FA si impegna a rispettare i tempi per l'effettuazione dei test di accettazione che devono essere eseguiti in maniera coerente con la programmazione dello sviluppo del sistema;
- la PA/FA deve richiedere che al termine di ogni iterazione/sprint il team faccia un esame di cosa ha funzionato e cosa no a scopo di miglioramento continuo;
- la ditta si impegna a fare, ogni mattina lavorativa, un briefing all'in piedi, di non più di 15 minuti, sul da farsi in quel giorno;
- la ditta si impegna a sviluppare il SW scrivendo, prima di passare alla fase di codifica della soluzione, i test che il codice sviluppato deve soddisfare perché la soluzione sia stata correttamente implementata (test-first development);
- la PA/FA si impegna a fornire a tempo pieno tutti gli esperti di sistema che devono operare insieme al team di sviluppo (diventano parte del team di sviluppo);
- la ditta si impegna a far eseguire la programmazione a coppie che consente ai Junior di imparare dai Senior.

In sintesi, la PA/FA potrebbe, senza grossi sforzi, includere, all'interno dei propri contratti, le pratiche agili illustrate sopra allo scopo di vincolare la ditta e la PA/FA stessa a rispettare il paradigma Agile che in ultima analisi agevolerebbe, in termini di conseguimento degli obiettivi, tanto la ditta fornitrice che la PA/FA committente senza danno per nessuno.