Lezione #21

MartedÃ¬, 26 *Novembre* 2013

Tullio Vardanega, 09:30-11:15

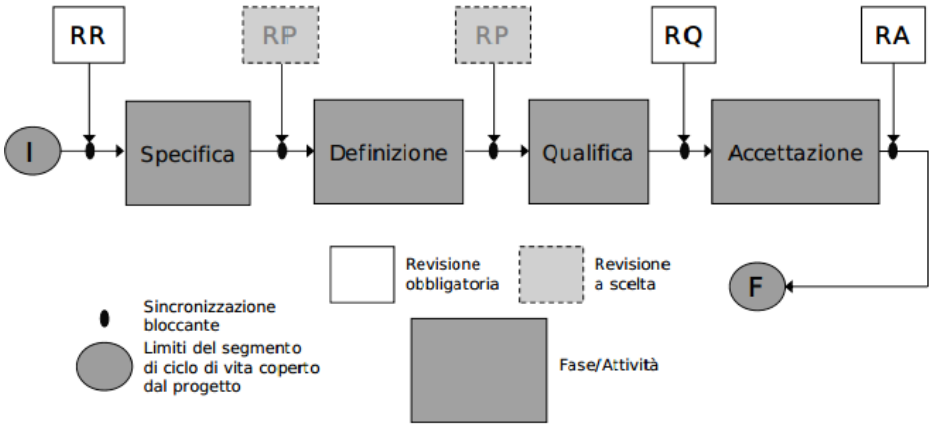
Luca De Franceschi

Un verificatore non agisce mai in fase di *correzione*, ma deve esistere una norma per la gestione delle modifiche, la correzione deve essere organizzata. Il verificatore non Ã¨ il possessore del documento.

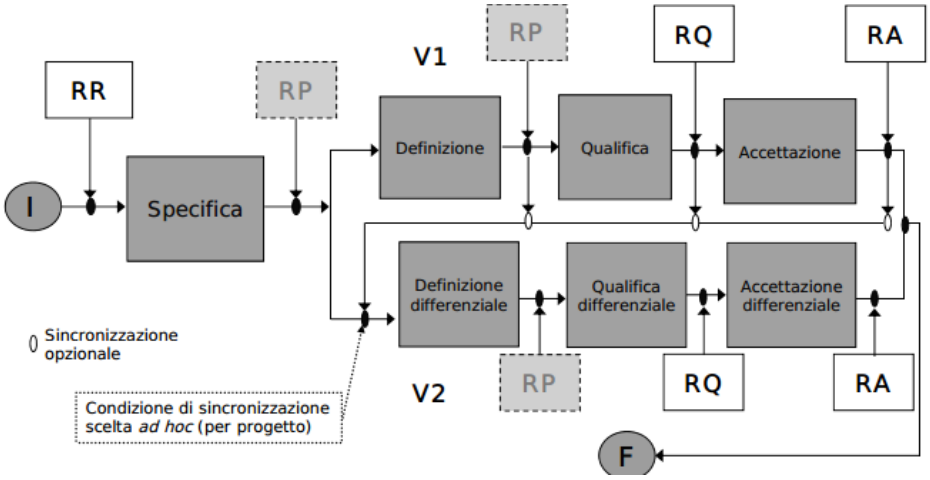
C'Ã¨ una distanza che va gestita tra l'ordine sequenziale delle cose che si aspetta il committente (fase esterna in cui si avanza linearmente, che Ã¨ il senso delle revisioni) e il flusso con cui interagiamo noi e il nostro modo di lavorare (che presumibilmente non sarÃ sequenziale). Per il committente perÃ²*nonÃ¨positivoavereunmodellosequenziale, p*

- Vi sono componenti di impegno **non comprimibili**, ovvero non posso mettere piÃ¹*persone sullostessocompito, nonposs*
- La verifica a livello di sistema si fa **solo alla fine**, perchÃ¨ non sono test parallelizzabili e il sistema diventa disponibile solo alla fine dello sviluppo.

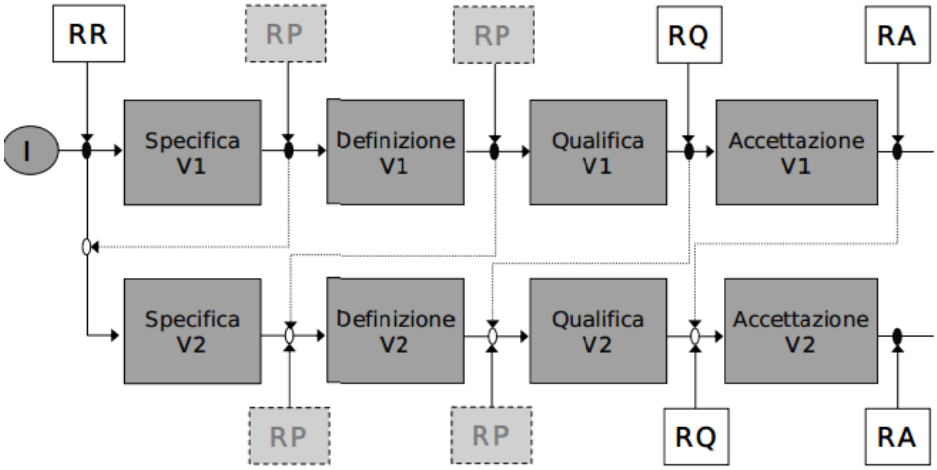
Occorre avere una pianificazione che abbia margini e che sia completamente consapevole dei vincoli. Una buona progettazione consente di non cadere nella iterazione non controllata. In questo modo, con queste tecniche si migliora la *mitigazione dei rischi*.
Che tipo di modello di ciclo di vita devo scegliere? Dobbiamo scegliere una strategia che sia buona per noi. Il modello sequenziale Ã¨ esattamente coerente con quello che si aspetta il committente, ma non per quanto riguarda il proponente.



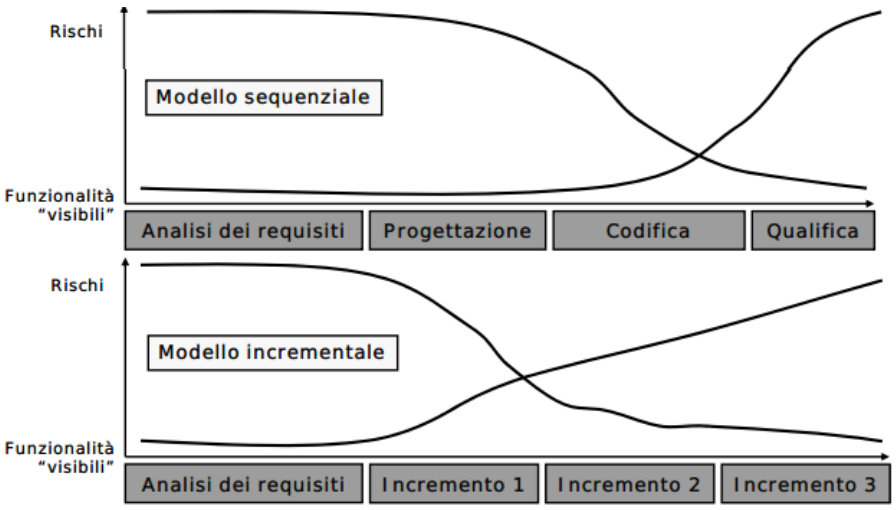
Il **modello incrementale** non Ã¨ iterativo, perchÃ¨ l'iterazione *distrugge* per sostituzione e quindi Ã¨ potenzialmente pericolosa. La posso avere solo in situazioni di emergenza. Un ciclo incrementale puÃ²*esserevistocomeun"for"*



Posso fare un modello in stile *Chrome* ("un numero grande di versioni Ã¨ bello"), **modello evolutivo**. E' un modello che approssima la soluzione finale ammettendo tante iterazioni, obortendo le versioni intermedie. Per poter attuare un modello evolutivo ho bisogno di tanta energia. E' una tecnica molto interessante ma con un enorme costo.



Gli utenti non portano innovazione ma vengono presi dall'onda dell'innovazione. Nel nostro progetto non Ã¨ una scelta molto agevole.
Il **modello agile** non Ã¨ facilmente rappresentabile. Si ragiona sulle cose da fare (**backlog**). Fra le cose da fare in un modello agile le persone prendono liberamente quello che faranno (ciascuno pesca un post-it a seconda del proprio estro). In un modello agile l'essenziale Ã¨ che per ogni cosa fatta l'effetto sia visibile (**incremental build**). Ogni aggiunta rende il prodotto sempre piÃ¹*vicinoalleaspettative*, *anchesenonhaunordineparticolarn*



Avvertenze. Quando si Ã¨ in progettazione bisogna avere la maturitÃ di capire che essa deve condurre per mano lo sviluppo, deve essere la soluzione, e dev'essere verificabile in modo piÃ¹*omenoautomatizzato.Nontuttiiproblemihann*

- **Obiettivi**;
- **Vincoli**;
- **Alternative**;
- **Rappresentazione del problema e delle sue soluzioni**.

La prima cosa da fare Ã¨ **decomporre** in modo modulare e senza dipendenze. Una buona decomposizione identifica componenti tra loro indipendenti. Le cose che decido localmente non devono avere un impatto pesante sul globale. La seconda cosa che voglio fare Ã¨ esporre all'esterno solo ciÃ²*cheÃ¨necessariosapere*(**incapsulamento**).*Vo*