Ingegneria del Software

Sviluppo rapido del software

Antonino Staiano
e-mail: **antonino.staiano@uniparthenope.it**

Scopo della lezione

- Spiegare come un processo di sviluppo iterativo, incrementale porta ad una consegna può veloce di software più utile
- Discutere l'essenza dei metodi di sviluppo agile
- Spiegare i principi e la pratica della programmazione estrema
- Spiegare i ruoli della prototipizzazione nel processo del software

Sviluppo software rapido

- A causa degli ambienti in cui operano le aziende che cambiano rapidamente, le stesse devono rispondere a nuove opportunità e competizioni
- Ciò richiede il software ed uno sviluppo veloce e la consegna non è spesso il requisito più critico per i sistemi software
- Le aziende possono accettare software di minor qualità se è possibile una rapida consegna del software dotato delle funzionalità essenziali

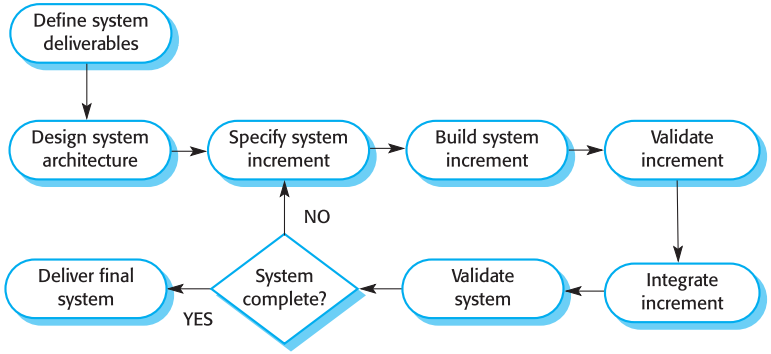
Requisiti

- A causa degli ambienti mutevoli, spesso è impossibile arrivare ad un insieme consistente e stabile di requisiti di sistema
- Pertanto, un modello a cascata per lo sviluppo non è praticabile e un approccio allo sviluppo basato su specifiche e consegne iterative è l'unico modo per consegnare il software velocemente

Caratteristiche dei processi RAD

- I processi di specifica, progettazione e implementazione sono concorrenti. Non c'è una specifica dettagliata e la documentazione associata la progetto è minimizzata
- Il sistema è sviluppato in una serie di incrementi. Gli utenti finali valutano ogni incremento e fanno proposte per gli incrementi successivi
- Le interfacce utente del sistema sono solitamente sviluppate usando un sistema di sviluppo interattivo

Un processo di sviluppo iterativo



Vantaggi dello sviluppo rapido

- Accelerazione nella consegna dei servizi del cliente. Ogni incremento consegna le funzionalità di maggiore priorità al cliente
- Partecipazione dell'utente nel sistema. Gli utenti devono essere coinvolti nello sviluppo il che si traduce nel fatto che il sistema ha una maggiore possibilità di soddisfare i loro requisiti e gli utenti sono più impegnati col sistema

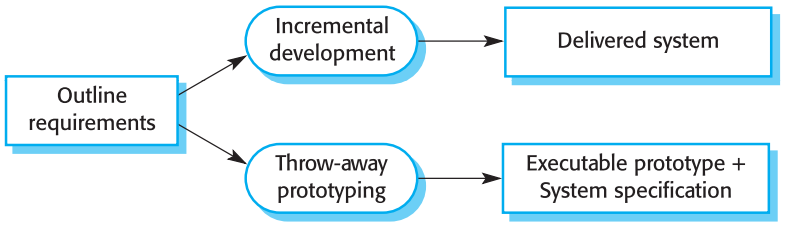
Problemi con lo sviluppo incrementale

- Problemi di gestione
  - Può essere difficile giudicare i progressi e complicato trovare i problemi poiché non c'è documentazione per illustrare cosa è stato fatto
- Problemi contrattuali
  - Il contratto tipico può includere una specifica; senza una specifica, devono essere usate diverse forme di contratto
- Problemi di convalida
  - Senza una specifica, rispetto a cosa si deve testare il sistema?
- Problemi di manutenzione
  - Le continue modifiche tendono a corrompere la struttura del software rendendolo più costoso rispetto ai cambiamenti e alle evoluzioni per soddisfare I nuovi requisiti

Prototipizzazione

- Per taluni grandi sistemi, lo sviluppo e la consegna iterativa incrementale può essere impraticabile; ciò è specialmente vero quando team multipli lavorano in luoghi differenti
- Può essere usata la prototipizzazione, in cui un sistema sperimentale è sviluppato come base per formulare i requisiti. Tale sistema è abbandonato quando si raggiunge l'accordo sulla specifica del sistema

Sviluppo incrementale e prototipizzazione



Conflitto di obiettivi

- L'obiettivo dello sviluppo incrementale è consegnare un sistema funzionante agli utenti finali. Lo sviluppo parte con i requisiti meglio capiti
- L'obiettivo della prototipizzazione è di convalidare o determinare i requisiti di sistema. Il processo di prototipizzazione inizia con i requisiti meno compresi

Metodi agili

- L'insoddisfazione con l'overhead introdotto nei metodi di progettazione ha portato alla creazione dei metodi agili. Questi metodi:
  - Mettono a fuoco il codice piuttosto che il progetto
  - Sono basati su un approccio iterativo allo sviluppo del software
  - Sono intesi a consegnare software funzionante velocemente e ad evolverlo velocemente per soddisfare i requisiti che cambiano
- I metodi agili sono probabilmente più adatti a sistemi aziendali di piccole/medie dimensioni o prodotti per PC

Principi dei metodi agili

Principio	Descrizione
Coinvolgimento del cliente	I clienti devono essere strettamente coinvolti in tutto il processo di sviluppo. Il loro ruolo è fornire nuovi requisiti, dar loro priorità e valutare i cicli del sistema
Consegna incrementale	Il software viene sviluppato per incrementi e il cliente specifica i requisiti da includere in ogni incremento
Persone, non processi	Le capacità del team di sviluppo devono essere riconosciute e sfruttate. I membri del team devono essere lasciati liberi di sviluppare con i propri metodi di lavoro senza processi prescrittivi
Accettare i cambiamenti	Prevedere i requisiti di sistema che cambieranno, quindi progettare il sistema in modo che possa accettare tali cambiamenti
Mantenere la semplicità	Concentrarsi sulla semplicità sia nel software che si sviluppa, sia nel processo di sviluppo. Quando possibile lavorare attivamente per eliminare le complessità del sistema

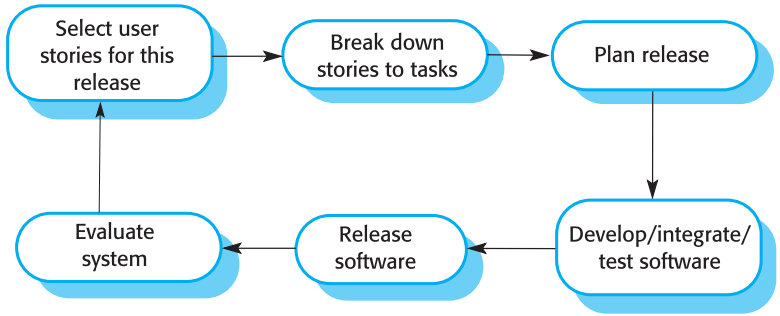
Problemi con i metodi agili

- Può essere difficile mantenere l'interesse dei clienti coinvolti nel processo
- I membri del team possono essere non adatti all'intenso coinvolgimento che caratterizza i metodi agili
- Priorizzare i cambiamenti può essere difficile laddove ci sono diversi stakeholder
- Mantenere la semplicità richiede un lavoro extra
- I contratti possono essere un problema così come per altri approcci allo sviluppo iterativo

Programmazione estrema

- Forse il metodo agile più noto e più diffusamente utilizzato
- La programmazione estrema (XP) impiega un approccio "estremo" per lo sviluppo iterativo
  - Possono essere costruite più volte ogni giorno nuove versioni
  - Gli incrementi sono consegnati ai clienti ogni due settimane
  - Tutti i test devono essere eseguiti per ogni versione e la versione è accettata solo se i test sono eseguiti con successo

Il ciclo di rilascio della XP



Pratiche di XP 1

Pianificazione incrementale	I requisiti sono registrati su Story Card incluse in una release, e sono determinate dal tempo disponibile e dalla loro priorità relative. Gli sviluppatori suddividono queste Storie in "Task" di sviluppo
Piccole release	Deve essere inizialmente sviluppato un insieme minimo di funzionalità che fornisce valore aziendale. Le release del sistema sono frequenti e aggiungono, in modo incrementale, funzionalità alla release iniziale
Progettazione semplice	Deve essere eseguita la progettazione sufficiente a soddisfare i requisiti attuali e niente di più
Sviluppo con test iniziale	Viene utilizzato un sistema automatico di test delle unità per testare una nuova parte di funzionalità prima che questa sia implementata
Refactoring	Tutti gli sviluppatori effettuano in continuazione il refactoring del codice appena si trovano miglioramenti al codice stesso. Questo rende il codice semplice e mantenibile

Pratiche di XP 2

Programmazione a coppie	Gli sviluppatori lavorano a coppie, verificando reciprocamente il lavoro e fornendo il supporto per fare sempre un buon lavoro
Possesso collettivo	Le coppie di programmatori lavorano su tutte le aree del sistema, in modo che non ci siano esperti isolate che sviluppano, ogni sviluppatore possiede tutto il codice, chiunque può cambiare qualsiasi cosa
Integrazione continua	Appena un task è concluso, viene integrato nel sistema. Dopo ogni integrazione tutti i test sulle unità del sistema devono essere superati
Ritmo sostenibile	Non sono considerati accettabili grandi ritardi, poiché spesso l'effetto finale è la riduzione della qualità del codice e della produttività a medio termine
Clienti on-site	Dovrebbe essere disponibile a tempo pieno una rappresentativa dell'utente finale del sistema (il Cliente) all'interno del team XP. In un processo di programmazione estrema, il cliente è un membro del team e ha la responsabilità di consegnare i requisiti del sistema al team di implementazione

XP e i principi agili

- Lo sviluppo incrementale è supportato attraverso il rilascio frequente di piccole release del sistema
- Il coinvolgimento del cliente comporta una partecipazione a tempo pieno con il team
- Le persone, non il processo, sono supportate dalla programmazione in coppia, dal possesso collettivo del codice e del sistema, e da un processo che non richieda periodi di lavoro eccessivamente lunghi
- Le modifiche sono supportate da release costanti del sistema
- Il mantenimento della semplicità è supportato dal costante refactoring del codice

Scenari dei requisiti

- Nella XP, i requisiti dell'utente sono espressi come scenari o storie dell'utente
- Sono scritte su schede ed il team di sviluppo le spezza in task di implementazione. Questi task costituiscono la base per la stima del piano di lavoro e dei costi
- Il cliente sceglie le storie da includere nella successiva release sulla base delle proprie priorità e le stime del piano di lavoro

Schede storiche per il download di documenti

Downloading and printing an article

First, you select the article that you want from a displayed list. You then have to tell the system how you will pay for it - this can either be through a subscription, through a company account or by credit card.

After this, you get a copyright form from the system to fill in and, when you have submitted this, the article you want is downloaded onto your computer.

You then choose a printer and a copy of the article is printed. You tell the system if printing has been successful.

If the article is a print-only article, you can't keep the PDF version so it is automatically deleted from your computer.

Modifiche dei requisiti e story card

- Se i requisiti variano
  - le storie non implementate possono essere cambiate o scartate
  - se le storie sono state implementate in un incremento già consegnato
    - Sono sviluppate nuove story card ed il cliente decide se queste modifiche devono avere la priorità sulle nuove funzionalità

XP: Estremizzazione dello sviluppo ciclico

- La XP sceglie un approccio estremo allo sviluppo critico
  - Nuove versioni del software possono essere costruite diverse volte al giorno e gli incrementi consegnati al cliente circa ogni due settimane
  - Quando un programmatore crea una nuova versione del sistema deve eseguire tutti i test automatici esistenti oltre ai test per le nuove funzionalità
  - Il software è accettato solo se supera con successo tutti i test

XP e cambiamento

- Il buon senso convenzionale nella pratica dell'ingegneria del software predica che la progettazione avvenga per favorire le modifiche
  - E' importante dedicare tempo e sforzi nel pianificare i possibili cambiamenti in modo da ridurre i costi nelle fasi successive del ciclo di vita
- XP, tuttavia, considera che questo punto non è fondamentale poiché i cambiamenti non possono essere anticipati in modo realistico
- Piuttosto, propone un costante miglioramento del codice (refactoring) per rendere semplici le modifiche quando devono essere implementate

Test in XP

- Rappresenta una delle principali differenze tra lo sviluppo ciclico e lo sviluppo pianificato
  - Nello sviluppo ciclico non c'è una specifica del sistema utilizzata da un team di test esterno
  - Alcuni approcci allo sviluppo ciclico hanno un processo di test molto informale
- La programmazione estrema dà più enfasi al processo di test rispetto ad altri metodi agili per evitare problemi di test e convalida

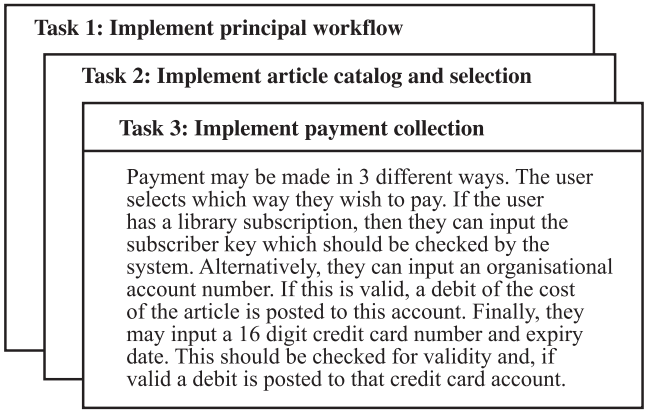
Test in XP

- Gli elementi caratteristici del test nella programmazione estrema
  - Sviluppo con test iniziale
  - Sviluppo incrementale dei test a partire dagli scenari
  - Coinvolgimento dell'utente nello sviluppo dei test e nella convalida
  - Uso di strutture automatiche per il test

Test iniziale

- Lo sviluppo del test iniziale è la maggiore innovazione
  - Scrivere prima i test definisce in modo implicito un'interfaccia e una specifica comportamentale per la funzionalità da sviluppare
    - Si riducono problemi dei requisiti e le incomprensioni dell'interfaccia
  - L'approccio è adatto nei processi che hanno una chiara relazione tra il requisito del sistema e il codice che lo implementa
    - In XP ciò è vero perché le story card sono suddivise in attività e le attività sono le unità principali dell'implementazione

Schede dei compiti per il download dei documenti



Creazione dei test

- I requisiti utente sono espressi sotto forma di scenari o storie, ordinati, dall'utente, secondo la loro priorità di realizzazione
  - Il team di sviluppo valuta ogni scenario e lo suddivide in task
  - Ogni task rappresenta una funzionalità per la quale si può progettare un test di unità
  - Ogni task, per verificare la propria implementazione, genera uno o più test unitari

Descrizione casi di test

Test 4: Test credit card validity
Input: A string representing the credit card number and two integers representing the month and year when the card expires **Tests:** Check that all bytes in the string are digits Check that the month lies between 1 and 12 and the year is greater than or equal to the current year. Using the first 4 digits of the credit card number, check that the card issuer is valid by looking up the card issuer table. Check credit card validity by submitting the card number and expiry date information to the card issuer **Output:** OK or error message indicating that the card is invalid

Sviluppo con test iniziale

- Scrivere i test prima del codice chiarisce i requisiti da implementare
- I test sono scritti come programmi piuttosto che dati in modo che essi possano essere eseguiti automaticamente. Il test include un controllo che sia eseguito correttamente
- Tutti i test precedenti e nuovi, sono eseguiti automaticamente quando sono aggiunte nuove funzionalità. In questo modo si controlla che le nuove funzionalità non abbiano introdotto errori

Qualche problema con il test iniziale

- I programmatori preferiscono la programmazione al testing
  - Scrivono test incompleti che non verificano situazioni insolite
  - Alcuni test possono essere difficili da scrivere
    - Interfaccia utente complessa
  - Pur avendo molti test del sistema, il proprio insieme di test può non fornire una copertura completa
    - Le parti cruciali potrebbero non essere eseguite e quindi non testate
  - Affidarsi al cliente è difficile
    - Poco tempo a disposizione e incapaci di lavorare con il team a tempo pieno
    - I clienti pensano che fornire i requisiti sia sufficiente. Riluttanti a farsi coinvolgere nel processo di test

Programmazione in coppia

- Nella XP, i programmatori lavorano in coppia nello sviluppo del codice
- Questo aiuta a sviluppare un possesso comune del codice e distribuisce la conoscenza attraverso il team
- Serve come un processo di revisione informale poiché ogni linea del codice è guardata da più di una persona
- Incoraggia il refactoring siccome l'intero team ne può beneficiare
- Studi empirici suggeriscono che la produttività nello sviluppo con la programmazione in coppia e simile a quella di due persone che lavorano in modo indipendente

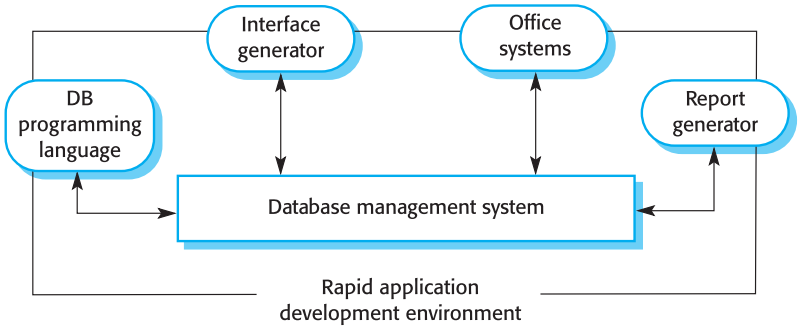
Sviluppo rapido di applicazioni (RAD)

- I metodi agili hanno ricevuto una notevole attenzione sebbene anche altri approcci alla sviluppo rapido di applicazioni siano stati usati per tanti anni
- Concepiti per applicazioni commerciali che utilizzano dati in modo massiccio e basati sulla programmazione e la presentazione di informazioni da un database

Strumenti per l'ambiente RAD

- Linguaggio di programmazione per database
- Generatore di interfaccia
- Collegamenti con applicazioni da ufficio
- Generatore di rapporti

Un ambiente RAD



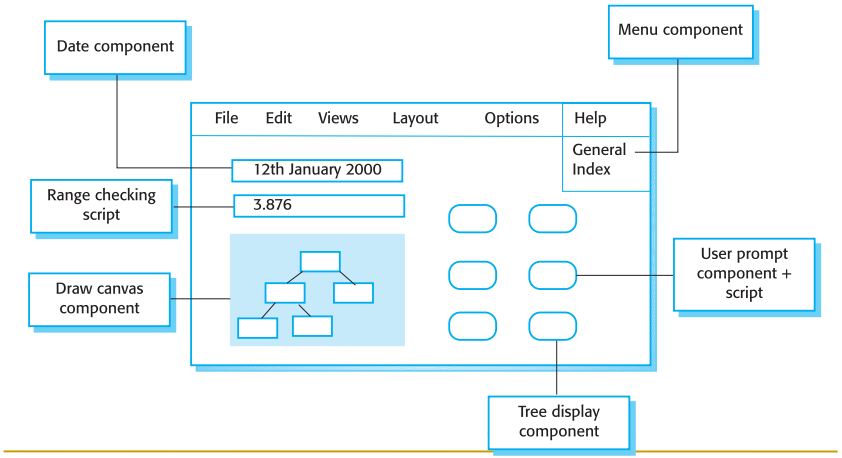
Generazione di interfaccia

- Molte applicazioni sono basate su molteplici *form* complesse. Sviluppare manualmente le *form* è un'attività molto costosa in termini di tempo
- Gli ambienti RAD includono il supporto per la generazione di schermate:
  - Definizione di *form* interattiva mediante tecniche *drag and drop*
  - Collegamenti tra *form* laddove ne è specificata la sequenza di presentazione
  - Verifica delle *form* laddove sono definiti campi del range dei valori consentiti

Programmazione visuale

- Linguaggi di scripting come Visual Basic supportano la programmazione visuale dove è sviluppato un prototipo creando un'interfaccia utente da oggetti standard ed associandovi dei componenti
- Esiste un'ampia gamma di librerie di componenti per supportare questo tipo di sviluppo
- Queste possono essere fatte su misura per adattarsi ai requisiti dell'applicazione specifica

Programmazione visuale con riuso



Problemi con lo sviluppo visuale

- Difficile coordinare lo sviluppo basato su un team
  - Soprattutto per sistemi complessi che coinvolgono grossi team di persone
- Non c'è un'architettura esplicita del sistema
- Parti di programma con dipendenze complesse possono creare problemi di manutenibilità

Riuso COTS (commercial off-the-shelf)

- Un approccio efficace allo sviluppo rapido consiste nel configurare e collegare sistemi esistenti
- Ad esempio, un sistema di gestione dei requisiti potrebbe essere costruito usando:
  - Un database per memorizzare i requisiti
  - Un elaboratore testi per individuare i requisiti e produrre report
  - Un foglio di calcolo per tabelle per la tracciabilità dei requisiti

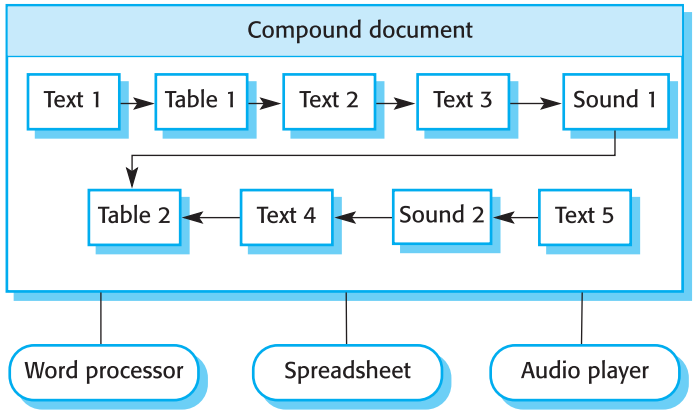
Documenti composti

- Per alcune applicazioni, può essere creato un prototipo sviluppando un documento composto
- E' un documento con elementi attivi (come un foglio di calcolo) che consente all'utente di fare calcoli
- Ogni elemento attivo ha un'applicazione associata che viene invocata quando l'elemento è selezionato
- Il documento stesso funge da integratore di differenti applicazioni

Documento composti (cont.)

- Vantaggi
  - Molte funzionalità dell'applicazione possono essere implementate velocemente a basso costo
  - Gli utenti che hanno già familiarità con le applicazioni che formano il sistema non devono imparare come utilizzare le nuove funzionalità
- Svantaggi
  - Se non capiscono le singole funzionalità può essere difficile impararle perché possono essere confuse con le funzionalità dell'applicazione non necessarie
  - Problemi di prestazioni con l'applicazione a causa di passare da un sistema applicativo ad un altro
    - L'overhead del passaggio dipende dal supporto fornito dal SO

Collegamento di applicazioni



Prototipizzazione del software

- Un prototipo è una versione iniziale di un sistema usato per dimostrare i concetti e provare le opzioni di progettazione
- Un prototipo può essere usato nel
  - Processo di ingegneria dei requisiti per aiutare la scoperta e la convalida dei requisiti
  - Nei processi di progettazione per esplorare le opzioni e sviluppare un progetto per le interfacce utente
  - Nel processo di test per eseguire test back-to-back

Impiego dei prototipi

- I prototipi permettono agli utenti di vedere come il sistema supporta il loro lavoro
  - Possono suggerire nuove idee per i requisiti e favorire la scoperta di punti di forza e debolezze del sistema
- Un prototipo può essere utilizzato mentre si sta sviluppando il sistema
  - per eseguire esperimenti di progettazione
  - per verificare la validità di un progetto proposto
  - Es: un progetto per un DB può essere prototipato e testato per verificare un accesso efficiente ai dati per le query più comuni
  - Progettazione interfacce grafiche: descrizioni testuali e diagrammi non sono efficaci per i requisiti delle GUI

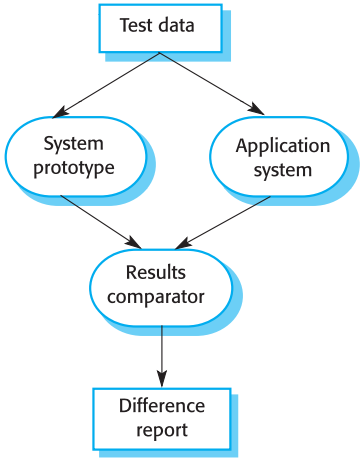
Impiego dei prototipi (cont.)

- Nel testing, uno dei principali problemi è la convalida del test
  - Si deve verificare che i risultati di un test siano quelli previsti
- Con un prototipo
  - Si riduce lo sforzo di verifica dei risultati con un test back-to-back
    - Prototipo e sistema sono sottoposti agli stessi testcase
      - Se forniscono gli stesi risultati , probabilmente il test non ha individuato errori
      - Se i rusltati sono diversi, può esserci un errore nel sistema e bisogna indagare le cause della diversità
- Riducono il tempo impiegato per lo sviluppo della documentazione utente e per esercitare gli utenti nell'uso del sistema

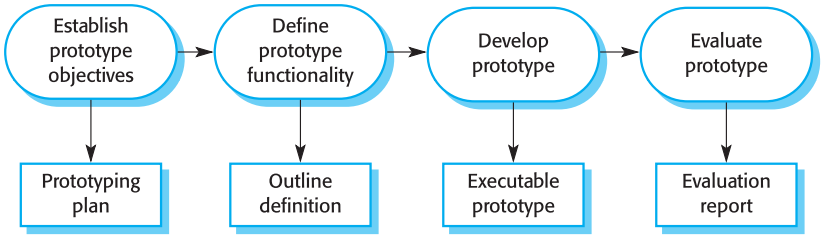
Benefici della prototipizzazione

- Migliore usabilità del sistema
- Un'aderenza più stretta con le reali necessità degli utenti
- Migliore qualità del progetto
- Migliore manutenibilità
- Sforzi per lo sviluppo ridotti

Test back-to-back



Processo di prototipizzazione



Prototipi throw-away

- I prototipi dovrebbero essere scartati dopo lo sviluppo in quanto non costituiscono una buona base per lo sviluppo del sistema:
  - Può essere impossibile raffinare il sistema per soddisfare i requisiti non funzionali
  - I prototipi normalmente non sono documentati
  - La struttura del prototipo si degrada durante i cambiamenti rapidi che si susseguono
  - Il prototipo probabilmente e non soddisferà i normali standard di qualità dell'organizzazione