- ***Lezione 7 – (31/10/2024)***

- Il Modello Netscape

M. A. Cusumano e D. B. Yoffie hanno fatto un'esperienza simile, a quella che *M. A. Cusumano e R. W. Selby* hanno fatto alla Microsoft, in un'azienda più piccola dal nome Netscape, la quale lavorava a progetti adattabili allo sviluppo di applicazioni Internet, browser e prodotti server. Anch'essi, nell'Ottobre del 1999, pubblicarono un articolo sulla rivista *IEEE Computer*, l'articolo *Software Development on Internet Time*.

La Netscape adottava lo stesso modello *synchronize-and-stabilize* che adottava la Microsoft.

Dal punto di vista della produttività i risultati sono simili tra le due aziende, nonostante la Netscape aveva una dimensione dello staff differente. Infatti aveva in media 1 tester ogni 3 sviluppatori.

Loro hanno osservato nell'azienda i seguenti difetti:

- Scarso effort di pianificazione (tranne che su prodotti server)
- Documentazione incompleta
- Scarso controllo sullo stato di avanzamento del progetto (lasciato all'esperienza e all'influenza dei project manager)
- Scarso controllo su attività di ispezione del codice (codereview)
- Pochi dati storici per il supporto alle decisioni

Nell'articolo era presente la seguente tabella dove si vede che l'azienda era veramente piccola

Table 2. Allocation of staff in Netscape's client and server development divisions, mid-1998.

	Client products	Server products	Total
Software engineering	110	200	310
Testing (QA)	50	80	130
Product management	50	42	92
Subtotal	210	322	533
Other*	30	98	128
Total	240	420	660

*Persons in activities such as documentation, user interface design, OEM porting, internationalization and localization, and special product support.

Il processo di sviluppo Netscape:

- **Step 1: Requisiti del prodotto e proposta del progetto**

    - Pianificazione anticipata del meeting (APM) tenuta per fare brainstorming di idee (marketing, sviluppo, esecutivo).
    - Si sviluppa la visione del prodotto, inizialmente dagli ingegneri più esperti, adesso principalmente dai responsabili di prodotto (product managers).
    - Della progettazione e scrittura di codice dagli ingegneri per esplorare tecnologie alternative o idee tecniche.
    - Documento dei requisiti del prodotto compilato dai responsabili del prodotto con l'aiuto degli sviluppatori.
    - Revisione non ufficiale di queste specifiche preliminari dagli ingegneri.
    - La specifica funzionale inizia dagli ingegneri, talvolta con l'aiuto dai responsabili del prodotto.
    - Programma e piano di bilancio compilati dal settore commerciale ed ingegneristico e discusso in modo non ufficiale con i dirigenti.

- **Step 2: Prima revisione esecutiva**

    - Documento di revisione esecutiva dei requisiti del prodotto, programma e proposta di bilancio;
    - Piano modificato secondo le necessità;

- **Step 3: Inizio della fase di sviluppo**

    - Caratteristiche di design e scrittura in codice, lavoro di progettazione secondo le necessità;
    - Integrazione giornaliera dei componenti come sono creati e costruiti;
    - Lista dei problemi generati e inizio delle correzioni;

- **Step 4: Revisione esecutiva provvisoria (se necessario)**

    - Le specifiche funzionali dovrebbero essere complete a questo punto;
    - Correzione di metà strada nelle specifiche o risorse del progetto dove necessario;
    - Problemi di coordinamento con altri prodotti o progetti discussi dove necessario;
    - Continuo dello sviluppo;

- **Step 5: Primo rilascio interno (alpha) (impiega approssimativamente sei settimane)**

    - Lo sviluppo si arresta temporaneamente.
    - Intensivo debugging e testing del codice esistente;
    - Rilascio dell'Alpha per un feedback interno (o possibilmente un rilascio dello sviluppatore);
    - Continua lo sviluppo;
    - Inserimento del feedback dell'utente;
    - Obiettivo completamento delle funzionalità (raramente conseguito, nonostante i server in particolare cercano di essere il più completi possibile);
    - Una settimana per stabilizzare il rilascio beta;

- **Step 6: Pubblicazione di beta 1 o test sul campo 1 (impiega approssimativamente sei settimane)**

    - Ripetizione dei passi dello sviluppo e dei test nello step 5;
    - Gruppi server passano a test sul campo con clientela limitata piuttosto che beta pubbliche;

- **Step 7: Pubblicazione beta 2 e 3 (ogni beta impiega all'incirca sei settimane)**

    - Ripetizione dei passi dello sviluppo come nello Step 5;
    - Congelamento dell'interfaccia utente milestone (UI freeze milestone);
    - Completamento della funzione dello status "mandatorio", anche se sono ancora concessi piccoli cambiamenti;

- **Step 8: Completamento del codice**

    - Nessuna aggiunta di codice eccetto quello per correggere i bug; le caratteristiche sono funzionalmente complete;

- **Step 9: Prova finale e rilascio**

    - Ricerca degli errori (debugging) finale e stabilizzazione della versione di rilascio;
    - Meetings di abilitazione con i dirigenti più esperti (senior) per la decisione del lancio;
    - Rilascio per la produzione (RTM Release to manufacturing) e rilascio commerciale;

All’inizio degli anni 2000, è emerso un approccio critico verso i processi software tradizionali, spesso percepiti come troppo rigidi e limitanti per i team di sviluppo. Questo ha portato all’introduzione dei *metodi agili,* un insieme di pratiche che estendevano il concetto di sviluppo iterativo e incrementale, privilegiando aspetti come la comunicazione continua, il feedback rapido e la flessibilità nel modo di lavorare.

Il *Manifesto Agile*, documento fondamentale per questo movimento, raccoglie i valori e i principi condivisi dai metodi agili. I valori principali del *Manifesto Agile* sono:

- Valorizzare le persone e le interazioni più dei processi e degli strumenti.

- Preferire un software funzionante rispetto alla documentazione esaustiva.

- Favorire la collaborazione con il cliente invece di una rigida negoziazione contrattuale.

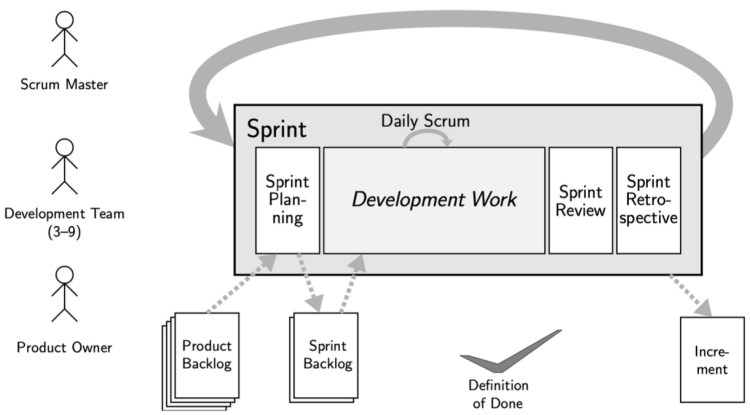
- Scegliere di adattarsi ai cambiamenti piuttosto che seguire un piano fisso.

Questi valori non eliminano del tutto gli elementi più tradizionali, ma stabiliscono che il team darà priorità agli aspetti dinamici e flessibili del lavoro.

In aggiunta ai valori, il *Manifesto Agile* descrive dodici principi che offrono ulteriori indicazioni per applicare correttamente i metodi agili, promuovendo un approccio collaborativo e adattabile nello sviluppo del software. Insieme, questi valori e principi formano la base di ciò che oggi chiamiamo *sviluppo agile.*

Quello che noto oggi giorno da aziende che sviluppano software è l’approccio *Scrum* (tradotto *“mischia”*). Questo approccio non ne stato introdotto per lo scopo dell’ingegneria del software, è sato introdotto da due giapponesi, ben prima del *Manifesto Agile.*

La metodologia è formalizzata nella *Scrum Guide*, che descrive ruoli, eventi e artefatti principali.



La metodologia prevede tre ruoli principali:

1. *Scrum Master* → Facilita l'implementazione di Scrum, assicurando che il team e il proprietario del prodotto rispettino le regole e aiutando gli stakeholder a interagire efficacemente con il team.

2. *Proprietario del Prodotto* → Gestisce il backlog del prodotto, dando priorità ai requisiti da sviluppare per garantire che le funzionalità più importanti siano affrontate prima.

3. *Team di Sviluppo* → Responsabile dello sviluppo del prodotto, svolge attività che includono progettazione, codifica e test, collaborando per realizzare incrementi di prodotto funzionanti.

Il lavoro viene organizzato in cicli chiamati *Sprint*, della durata di 2-4 settimane. Durante ciascun *Sprint*, il team lavora su un insieme predefinito di elementi scelti dal backlog del prodotto. Ogni Sprint segue un ciclo strutturato:

1. *Sprint Planning* - Inizio dello Sprint, in cui il team definisce gli obiettivi selezionando gli elementi da sviluppare e trasferendoli nel backlog dello Sprint.

2. *Daily Scrum* - Riunione giornaliera breve (stand-up) in cui il team si allinea sulle attività e affronta eventuali ostacoli.

3. *Sprint Review* - Alla fine dello Sprint, l'incremento sviluppato viene presentato al proprietario del prodotto e agli stakeholder per ottenere feedback.

4. *Sprint Retrospective* - Riunione finale per analizzare le performance e individuare possibili miglioramenti per i successivi Sprint.

Ogni *Sprint* produce un incremento di software funzionante e migliorato rispetto alla versione precedente, seguendo un processo che incoraggia la collaborazione, la trasparenza e l'adattabilità del team.

Un altro elemento fondamentale di *Scrum* è la *Definition of Done*, che rappresenta il criterio di completezza stabilito dal team di sviluppo per considerare un'attività realmente conclusa. La *Definition of Done* prevede che il lavoro sia sufficientemente testato e integrato, senza introdurre errori nel codice principale.

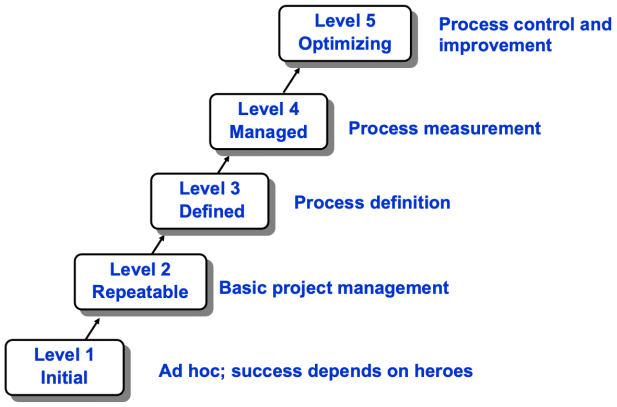
Un'altra pratica comune, spesso usata in combinazione con *Scrum* (ma non formalmente definita nella *Scrum Guide*), è quella delle *User Stories*. Le *User Stories* sono brevi descrizioni dei requisiti dell'utente, formulate come "storie" dal suo punto di vista. Usano un formato standard:

As a <role>, I want <goal> so that <benefit>

Le *User Stories* aiutano il team a comprendere le esigenze degli utenti. Quando una *User Story* è troppo grande viene suddivisa in *User Stories* più piccole per facilitarne la gestione e viene chiamata *Epics*.
Questi strumenti aiutano il team a mantenere chiarezza sugli obiettivi e sulla qualità del lavoro completato, migliorando la gestione dei requisiti e l'efficienza dei cicli di sviluppo.

- Modello di Maturità delle Capacità (CMM Capability Maturity Model):

  Il *Capability Maturity Model (CMM)*, introdotto nel 1993 dal *SEI (Software Engineering Institute)*, è un modello ideato per valutare il livello di maturità del processo software all'interno di un'organizzazione, misurando l'efficacia complessiva nell'applicazione delle tecniche di ingegneria del software. Questo modello utilizza un questionario e uno schema di valutazione organizzato su cinque livelli gerarchici; ogni livello include tutte le caratteristiche definite nei livelli precedenti, rendendolo un modello di tipo "additivo". Ciò significa che, per raggiungere un determinato livello di maturità, è necessario aver soddisfatto i requisiti di tutti i livelli precedenti.



Noi illustreremo il modello *CMM originale*, senza considerare le evoluzioni successive, come il *CMMI (Capability Maturity Model Integration),* in cui la *"I"* finale indica appunto l'integrazione *(Integrated).*

Ecco i cinque livelli di maturità del modello CMM:

- *Il livello Inziale* è un livello dove non esiste ancora un processo software strutturato; il processo viene sviluppato in modo ad hoc per ciascun progetto. Il successo è quindi fortemente legato all'impegno e alle competenze delle singole persone coinvolte nello sviluppo del software, piuttosto che a processi stabiliti.

- Il secondo livello, chiamato *Repetable,* qui sono presenti processi di base per gestire efficacemente i progetti, permettendo di monitorare costi, pianificazioni e attività. L'obiettivo è quello di replicare i successi ottenuti nei progetti

passati. Vengono applicate tecniche di project management di base, e il livello prende il nome proprio dalla capacità di ripetere pratiche efficaci.

- Il terzo livello, chiamato *Defined*, qui il progetto è definito sia da un punto di vista gestionale che ingegneristico. È qui che la maggior parte delle aziende riesce a ottenere la certificazione, poiché il livello 3 è quello con il maggior numero di certificazioni rilasciate. I processi sono standardizzati e documentati, fornendo una solida base per lo sviluppo software.

- Il quarto livello, chiamato *Managed*, ovvero livello *"gestito",* qui dopo aver definito il progetto, l'organizzazione è in grado di misurare e analizzare l'efficacia del proprio processo di sviluppo, basandosi su dati quantitativi. Ciò permette di avere un controllo più accurato delle performance e della qualità dei processi software.

- Il quinto livello, chiamato *Optimizing*, qui dopo aver definito il processo ed essere in grado di misurarlo sono in grado di migliorarlo continuamente.

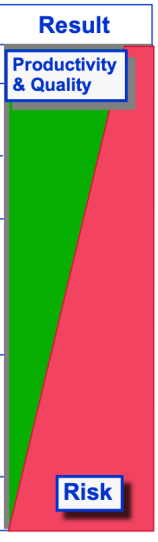
Per definire e valutare il livello di maturità di un'organizzazione secondo il modello CMM, il SEI ha individuato delle aree chiave, denominate *Key Process Areas (KPA)*. Ogni *KPA* rappresenta una funzione specifica che l'organizzazione deve implementare per conseguire la certificazione del livello desiderato. Complessivamente, esistono 18 *KPA* da soddisfare per poter raggiungere il quinto livello del modello CMM.

Ogni KPA è descritta rispetto a:

- Obiettivi
- impegni e responsabilità da assumere
- capacità e risorse necessarie per la realizzazione – attività da realizzare
- metodi di "monitoring" della realizzazione
- metodi di verifica della realizzazione

Ecco allocate le 18 KPA per ciascun livello

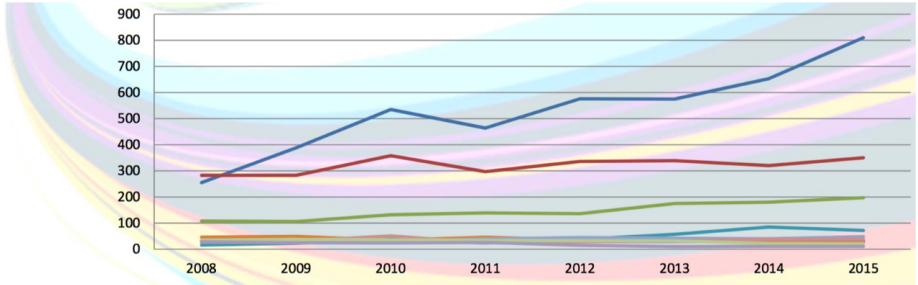
Level	Characteristic	Key Process Areas
Optimizing (5)	Continuous process capability improvement	Process change management Technology change management Defect prevention
Managed (4)	Product quality planning; tracking of measured software process	Software quality management Quantitative process management
Defined (3)	Software process defined and institutionalized to provide product quality control	Peer reviews Intergroup coordination Software product engineering Integrated software management Training program Organization process definition Organization process focus
Repeatable (2)	Management oversight and tracking project; stable planning and product baselines	Software configuration management Software quality assurance Software subcontract management Software project tracking & oversight Software project planning Requirements management
Initial (1)	Ad hoc (success depends on heroes)	"People"



Una Statistica fatta a febbraio 2000, che presenta una lista delle organizzazioni a livello 4 e 5 (maturità elevata) include:

- 71 organizzazioni negli USA, 44 a livello 4 (tra cui Oracle, NCR, Siemens Info Systems, IBM Global Services) e 27 organizzazioni a Livello 5 (tra cui Motorola, Lockeed-Martin, Booeing, Honeywell)
- 25 organizzazioni al di fuori degli USA, 1 a livello 4 in Australia, 14 a livello 4 in India, 10 a livello 5 in India

Number of appraisals by country (06/15)



	2008	2009	2010	2011	2012	2013	2014	2015
China	255	388	535	464	576	575	653	810
United States	283	283	358	297	336	339	320	350
India	108	106	132	139	136	175	180	197
Spain	30	30	30	30	30	30	30	30
Mexico	16	23	39	24	38	57	85	72
Japan	46	49	35	46	36	32	26	25
Korea, Republic Of	27	36	29	39	45	42	40	48
Brazil	28	32	51	25	28	33	38	38
France	35	37	36	34	33	32	22	21
United Kingdom	28	25	24	26	15	10	11	11

Trends (as of June 2015)