ROMA TRE UNIVERSITÀ DEGLI STUDI
Corso di Laurea in Matematica
Dipartimento di Matematica e Fisica

Sistemi per l'elaborazione delle informazioni

9. Principi di ingegneria del software

Dispense del corso IN530 a.a. 2019/2020

prof. Marco Liverani

Scopo dell'ingegneria del software

- L'ingegneria del software **definisce modelli e metodologie per formalizzare il processo di progettazione, realizzazione e manutenzione** di un sistema informatico
- In questo ambito si definisce un vero e proprio **ciclo di vita del software**, nel tentativo di trasformare lo sviluppo software da un'attività artigianale verso un processo industriale
  - in un'attività **artigianale** si possono raggiungere elevati livelli qualitativi e di performance, ma questo è strettamente legato all'**abilità dell'artigiano**; non sempre l'abilità può essere trasferita ad altri mediante un processo di formazione
  - in un'attività **industriale** si cerca di raggiungere un livello di qualità e di performance **adeguati al posizionamento di mercato del prodotto**; qualità e performance sono ottenuti mediante la definizione e la formalizzazione di **processi** e **modalità di realizzazione** di ciascuna attività necessaria nel ciclo di produzione; le attività sono **misurabili** e **controllabili**
- Oggi assume particolare importanza il tema del **Application Development Lifecycle Management** (ciclo di vita del processo di sviluppo software) e di tutte le attività che rientrano nell'ambito del ***Development Operations*** (**DevOps**)

Ciclo di vita del software

Un prodotto software ha un suo **ciclo di vita**, ossia esistono diverse fasi successive durante il processo di realizzazione e manutenzione del software che vanno dal suo concepimento iniziale, fino alla sua definitiva dismissione

1. **Analisi preliminare**
   - Studio di fattibilità, definizione delle esigenze, definizione del contesto
   - Definizione dei requisiti
2. **Progettazione del software**
   - Specifiche funzionali ed architetturali del software
   - Identificazione di librerie e componenti software riusabili
   - Progettazione delle componenti del sistema software
3. **Realizzazione del software**
   - Specifiche di dettaglio dei moduli
   - Sviluppo dei moduli software, test unitario
4. **Test, collaudo e rilascio in produzione**
   - Integrazione dei moduli e test unitari e di integrazione
   - Collaudo e validazione del sistema integrato
   - Rilascio, migrazione e caricamento dei dati, avvio in produzione
5. **Gestione del sistema software**
   - Utilizzo del software e manutenzione ordinaria, adeguativa, correttiva (MAC), evolutiva (MEV)
   - Rilascio di nuove release, ripetendo i passi 1–4
6. **Termine del ciclo di vita del software**
   - Messa fuori servizio del software, backup dei dati, migrazione dati e servizi su nuovo software

Prima fase: Analisi preliminare

- Attività principali:
  - Studio di fattibilità, definizione dell'esigenza e della soluzione informatica, definizione dell'ambito del software, identificazione degli *stakeholders* (utenti, altri sistemi, ecc.)
  - Definizione dei **requisiti**
- Output prodotti:
  - Nota tecnica o documento di "vision": contesto, esigenze e soluzioni, problematiche, disegno generale del sistema
  - Identificazione delle risorse necessarie (umane e strumentali) e stime di impegno
  - Documento dei requisiti (caratteristiche del prodotto software, cosa deve fare)
- Figure professionali coinvolte:
  - Project Manager, Account Manager
  - Analista
  - Software Architect, System Architect
  - Esperto del contesto di business
- Modalità operativa:
  - Interviste presso il cliente (assessment)
  - Analisi dei dati e dei sistemi informatici preesistenti
  - Identificazione di soluzioni di mercato, open source o presenti in azienda utili al progetto

Seconda fase: Progettazione del software

- Attività principali:
  - Definizione delle specifiche grafico-funzionali del software
  - Definizione delle specifiche di disegno architetturale (architettura fisica, architettura di rete, architettura logica e delle componenti)
  - Progettazione delle componenti del sistema software
- Output prodotti:
  - Documento di specifiche funzionali, specifiche sui dati
  - Documento di disegno architetturale, identificazione dei prodotti hardware e software e delle componenti riusabili
- Figure coinvolte:
  - Project Manager
  - Analista
  - Software Architect, System Architect
  - Specialista di prodotto
- Modalità operativa:
  - Lavoro *in-house*
  - Use case diagram, sequence diagram, component diagram, entity/relationship diagram
  - Studio di soluzioni di mercato, open source o presenti in azienda

Terza fase: Realizzazione del software

- Avviene anche in concomitanza con la progettazione (le due fasi possono essere concorrenti, con uno slittamento in avanti della fase di realizzazione); si svolge *in-house* o presso il cliente a seconda degli strumenti tecnologici impiegati e dei vincoli contrattuali
- Attività principali:
  - Specifiche di dettaglio dei moduli software
  - Progettazione dei moduli software (classi, funzioni, package e librerie)
  - Sviluppo dei moduli software, test unitario
- Output prodotti:
  - Specifiche tecniche per lo sviluppo dei singoli moduli software (il dettaglio dipende dalla complessità del sistema e dal livello di coinvolgimento degli sviluppatori nella progettazione)
  - Specifiche che descrivono la struttura ad oggetti del modulo o il modello relazionale dei dati
  - Software documentato, sotto controllo di configurazione
- Figure coinvolte:
  - Project Manager
  - Software Architect
  - Analista-Programmatore, Programmatore
- Modalità operativa:
  - Lavoro in-house o presso il cliente
  - Class diagram, flow chart, documentazione del codice (es.: Javadoc)
  - Uso di strumenti di sviluppo in base al tipo di tecnologia scelta
  - Uso di strumenti di versioning e di tracciamento delle modifiche (controllo di configurazione)
  - Test continuo dei moduli rilasciati nel repository di progetto (il software si compila?)

Quarta fase: Test, collaudo e rilascio in produzione

- Avviene al termine dello sviluppo di un modulo software auto-consistente e costituisce una *milestone* significativa per il progetto
- Attività principali:
  - Integrazione dei moduli e test unitari e di integrazione
  - Test del sistema integrato
  - Pianificazione del rilascio in produzione e predisposizione delle risorse necessarie (server, rete, ecc.)
  - Rilascio e messa in produzione
- Output prodotti:
  - Piano dei test e report di esecuzione dei test unitari e di integrazione
  - Piano dei test e report di esecuzione dei test di sistema
  - Piano di indirizzamento di rete
  - Sistema funzionante nell'ambiente di produzione identificato insieme al cliente
  - Manuale di configurazione e gestione
  - Manuale utente
  - Piano di backup, piano di *disaster recovery*
  - Istruzioni per il *change management* e il *porting* dei dati
- Figure coinvolte:
  - Project Manager
  - Software Architect
  - Analista-Programmatore, Programmatore
  - Sistemista
- Modalità operativa:
  - Lavoro presso il cliente nell'ambiente di produzione finale del prodotto (in alcuni casi nell'ambiente di collaudo, supportando poi il cliente per il passaggio in ambiente di produzione)

Quinta fase: Gestione e manutenzione del sistema in produzione

- Avviene dopo l'entrata in esercizio di almeno una delle componenti software significative; è un'attività priva di scadenze intermedie, ma con necessità di rendicontazione puntuale delle attività con una periodicità e un dettaglio definiti a livello contrattuale; la manutenzione correttiva è soggetta a SLA (*service level agreement*) definiti contrattualmente
- Attività principali:
  - Supporto agli utenti nell'utilizzo del sistema e formazione, supporto agli amministratori nella gestione del sistema
  - Supervisione del corretto funzionamento del sistema, presidio del sistema, backup/recovery dei dati
  - Manutenzione ordinaria, Manutenzione correttiva ed evolutiva
- Output prodotti:
  - Report di intervento
  - Piano di formazione
  - Documentazione delle modifiche software, con aggiornamento della documentazione
- Figure coinvolte:
  - Project Manager
  - Programmatore, Analista-Programmatore
  - Sistemista
  - Esperto di prodotto/di sistema
  - Operatore help desk
- Modalità operativa:
  - Lavoro presso il cliente nell'ambiente di produzione finale del prodotto per le attività di assistenza, gestione e formazione
  - Lavoro *in-house* e presso il cliente per le attività di manutenzione correttiva ed evolutiva

Sesta fase: Termine del ciclo di vita del sistema

- Attività principali:
  - Supporto agli amministratori per il trasferimento dei dati
  - Backup finale dei dati e del software
  - Supporto al cliente per la definizione di un processo di passaggio dal vecchio al nuovo sistema
- Output prodotti:
  - Istruzioni per il *change management* e il *porting* dei dati
- Figure coinvolte:
  - Project Manager, Account Manager
  - Analista
  - Sistemista
  - Esperto di prodotto/di sistema
- Modalità operativa:
  - Lavoro presso il cliente nell'ambiente di produzione finale del prodotto, in collaborazione con il team che si occupa di implementare il nuovo sistema informatico

Aspetti rilevanti per la progettazione del prodotto software

- A monte del processo di progettazione e sviluppo del software, devono essere presi in esame ed analizzati attentamente alcuni aspetti che condizionano il prodotto software stesso
  - **Scopo del progetto**
    - Contesto di business in cui si opera
    - Obiettivi del progetto
  - **Cliente, committente ed altri attori**
    - Descrizione del cliente, ossia colui che ci pagherà per il lavoro svolto
    - Descrizione del committente, ossia colui che ha richiesto il lavoro
    - Altri soggetti coinvolti: sponsor, altre aziende, esperti di vario genere coinvolti dal cliente o dal committente
  - **Utenti del prodotto software**
    - Utenti finali, possibilmente raggruppati per ruolo nell'ambito del contesto di business e identificando la competenza nel settore di business e in ambito informatico
    - Priorità degli utenti nella valutazione del nostro lavoro
    - Partecipazione degli utenti al progetto (definizione dei requisiti, fornitura di informazioni critiche per la riuscita del progetto, test, approvazione, ecc.)
    - Utenti addetti alla gestione del sistema (sistemisti, e altri addetti)

Aspetti rilevanti per la progettazione del prodotto software

- **Vincoli obbligatori**
  - Vincoli alla soluzione (vincoli tecnici sulle caratteristiche generali o sui prodotti da usare o non usare)
  - Ambiente per l'implementazione (piattaforma di base, framework o altri middleware da usare)
  - Applicazioni da integrare (altri sistemi preesistenti con cui il nostro software si deve integrare)
  - Prodotti di mercato o open source da utilizzare
  - Ambiente di lavoro in cui è impiegato il prodotto (descrizione tecnica o anche logistica o "culturale")
  - Vincoli temporali (vincoli di tempo dovuti a fattori esterni o opportunità che devono essere colte entro una determinata finestra temporale)
  - Vincoli di budget (la pianificazione delle attività, le risorse e il tempo impiegato devono essere definiti tenendo conto di questo vincolo)
- **Convenzioni sui nomi**
  - Definizioni e acronimi utilizzati nell'ambito del progetto
- **Fatti rilevanti e altre assunzioni**
  - *Fatti*: fattori che hanno impatto sul sistema, pur non costituendo dei vincoli o dei requisiti (regole di business, processi esterni al sistema, ma che possono avere impatto su di esso in determinate condizioni, ecc.)
  - *Assunzioni*: ipotesi verosimili che vengono assunte come veritiere e su cui si basano alcune delle scelte di progetto; devono essere condivise con il cliente e con il committente e non possono contraddire la realtà oggettiva (es.: mi aspetto che tutti gli utenti abbiano un PC con certe caratteristiche; mi aspetto che i *device* mobili possano essere continuamente connessi alla rete, ecc.)

Aspetti rilevanti per la progettazione del prodotto software

- **Ambito in cui viene eseguito il progetto**
  - La situazione attuale (come fanno ora, senza il nostro prodotto?)
  - Contesto in cui si inserisce il prodotto software
  - *Business event* che devono essere implementati/supportati dal prodotto software
- **Ambito del prodotto software**
  - Limiti di competenza del prodotto software (fino a dove deve supportare il processo di business e cosa non deve fare)
  - *Use case* (casi d'uso) del prodotto software, ossia azioni auto-consistenti da parte degli utenti o di altri sistemi che il prodotto deve implementare o supportare

Definizione dei requisiti

- I **requisiti** del prodotto descrivono **cosa** il sistema deve offrire (*dominio del problema*) e non **come** il sistema deve essere sviluppato (*dominio della soluzione*)
- La **corretta definizione dei requisiti** è una delle attività che determinano la **riuscita di un progetto** e che talvolta si tende erroneamente a sottovalutare
- Applicarsi nella definizione dei requisiti è fondamentale per darsi la possibilità di analizzare a fondo il contesto e le problematiche organizzative e tecniche con cui ci si dovrà misurare nell'ambito del progetto
- I modelli di ciclo di vita del software moderni hanno abbandonato l'idea che sia possibile identificare i requisiti di un sistema software a priori: tendono a privilegiare approcci **iterativi**
- I **requisiti** sono l'elemento di riscontro per verificare se il prodotto software è un **prodotto di qualità**; spesso i requisiti del prodotto e la verifica del recepimento dei requisiti sono un aspetto rilevante del contratto tra il committente e l'azienda che sviluppa il software
- Esistono diversi tipi di requisiti che possono essere definiti anche in momenti diversi nell'ambito di un progetto software:
  - **Requisiti funzionali**: cosa fa il sistema, come deve interagire con gli utenti o con altri sistemi
  - **Requisiti non funzionali**: dati, performance, affidabilità, ecc.
  - **Requisiti architetturali**: prodotti/piattaforme che si devono o non si devono usare, struttura del sistema, ecc.
  - **Requisiti di progetto**: tempo, budget, risorse, logistica, ecc.
  - **Driver di progetto**: obiettivi organizzativi, politici, opportunità di mercato, ecc.
  - Altri aspetti che caratterizzano il progetto

Definizione dei requisiti

- I requisiti devono essere riportati su uno o più documenti e devono poter essere identificati chiaramente e in modo sintetico
- Esempio di schema per la definizione di un requisito:
  - **ID del requisito**: codice identificativo univoco del requisito
  - **Tipo di requisito**: funzionale, non funzionale, ecc.
  - **Business event o Use case** che richiede il requisito
  - **Descrizione**: breve descrizione del requisito
  - **Motivazione**: giustificazione del requisito (cosa succede se non viene implementato?)
  - **Criterio di verifica**: descrizione del criterio con cui si potrà verificare se il requisito è soddisfatto oppure no
  - **Priorità**: priorità del requisito rispetto ad altri (indispensabile o superfluo? Lo ha richiesto esplicitamente il cliente o è solo un modo per migliorare il sistema?)
  - **Dipendenze**: identificativi di altri requisiti da cui dipende questo requisito
  - **Data/versione**: data di creazione o modifica del requisito
- I requisiti e le dipendenze tra i requisiti costituiscono un **grafo orientato** (aciclico!) di cui si deve tenere conto, quando si intende modificare uno dei requisiti del sistema
- Si deve poter **tracciare** i requisiti in ogni componente del progetto: codice, casi di test, …
- La **tracciabilità** è un aspetto di qualità di un progetto software fondamentale per molte attività: l'analisi degli impatti di un cambiamento di requisiti, la verifica della correttezza di un'implementazione, il test e il regression test

Definizione dei requisiti

- **Requisiti funzionali e sui dati**
  1. Requisiti funzionali (cosa deve fare il sistema?)
  2. Requisiti grafico-funzionali (come deve presentarsi o deve reagire il sistema? Quale stile deve adottare? Come viene garantita la coerenza dell'interfaccia utente?)
  3. Requisiti sui dati (un diagramma E/R, ma anche altri requisiti relativi al ciclo di vita dei dati, alla loro persistenza, alla quantità, alla codifica, ecc.)
- **Requisiti sull'usabilità**
  1. Requisiti sulla facilità d'uso
  2. Requisiti per l'accesso di utenti con disabilità o in condizioni logistiche disagevoli
  3. Requisiti per l'internazionalizzazione del prodotto (prodotto multilingua)
- **Requisiti sulle performance**
  1. Requisiti sulla rapidità di risposta e sulla latenza del sistema
  2. Requisiti di precisione e accuratezza (precisione numerica, precisione nelle conversioni di dati, ecc.)
  3. Affidabilità e disponibilità del sistema (quali livelli di servizio deve reggere? Per quanto tempo il sistema può non essere disponibile?)
  4. *Fault-tolerance* (condizioni di errore a cui il sistema deve poter resistere)
  5. Requisiti di capacità (capacità di memorizzazione, capacità di servire un elevato numero di client contemporanei, ecc.)
  6. Requisiti di scalabilità (capacità del sistema di crescere orizzontalmente o verticalmente per venire incontro alla crescita delle esigenze del cliente)

Definizione dei requisiti

- **Requisiti sulla sicurezza**
  1. Requisiti sul controllo degli accessi (chi è autorizzato ad accedere, a quali condizioni, con quali metodi di identificazione, ecc.)
  2. Requisiti sull'integrità dei dati
  3. Requisiti sulla privacy (quali dati non devono essere memorizzati, chi può accedere a determinati dati, ecc.)
  4. Requisiti di auditing (quali eventi devono essere tracciati nei log del sistema, secondo quale modalità tecnica, ecc.)
- **Requisiti legali**
  1. Requisiti sulle normative a cui deve aderire il sistema (es.: gestione delle utenze come stabilito dal DPR 196/2003)
  2. Requisiti sugli standard da adottare (formato dei dati, protocolli, ecc.)
- **Requisiti sulla documentazione e sulla formazione**
  1. Manuali e documenti da produrre, lingua della documentazione, formato della documentazione
  2. Requisiti sui sistemi di aiuto on-line per l'uso del sistema, requisiti sul materiale didattico a supporto della formazione

Definizione dell'architettura

- La descrizione dell'**architettura** è un aspetto essenziale nella progettazione del software, dal momento che ciò che viene prodotto è spesso un **sistema**, formato da diverse componenti che interagiscono fra di loro
- I requisiti che hanno un impatto diretto sull'architettura del sistema e che quindi concorrono alla sua definizione, possono essere raggruppati sulla base della seguente classificazione "FURPS":
  - **Funzionalità**: il sistema deve essere capace di fornire determinate funzioni
  - **Usabilità**: il sistema deve essere facilmente fruibile dall'utente finale
  - **Affidabilità** (*Reliability*): il sistema deve gestire e possibilmente "resistere" ad errori e *crash*
  - **Prestazioni**: il sistema deve avere performance adeguate
  - **Supportabilità**: il sistema deve essere tale da consentire una corretta gestione e manutenzione

Definizione dell'architettura

Per descrivere l'architettura del sistema è necessario adottare "viste" diverse, fra loro complementari

- **Vista logica**
  - organizzazione concettuale del software in termini di strati, sottosistemi, package, framework, classi e interfacce più importanti; in questa "vista" si riassumono anche le funzionalità offerte dalle componenti software e dai sottosistemi
  - Può mostrare gli scenari d'uso più importanti o più critici al fine di mettere in evidenza il ruolo di ciascuna componente logica del software
- **Vista dei processi**
  - Descrive il modello del progetto e l'organizzazione in processi
- **Vista di deployment**
  - Descrive il livello fisico con cui sarà pubblicato in ambiente di produzione il sistema software
- **Vista sui dati**
  - Descrive il modello di rappresentazione delle informazioni gestite dal sistema ed i flussi di trasmissione
- **Vista sulla sicurezza**
  - Autenticazione degli utenti, single sign-on, modello autorizzativo, cifratura, ecc.

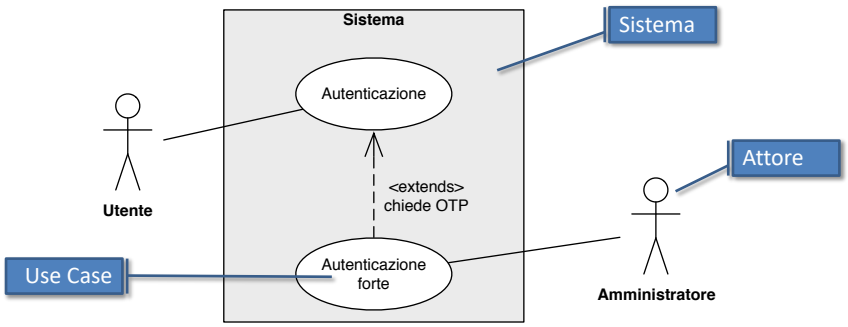
UML: Unified Modeling Language

- UML è un **formalismo grafico per descrivere e documentare l'architettura e le funzionalità di un sistema**; è un linguaggio di modellazione e specifica basato sul paradigma *object-oriented*
- UML svolge una funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti
- Il modello è strutturato secondo un insieme di viste che rappresentano diversi aspetti del sistema modellato (funzionamento, struttura, comportamento, ecc.)
- UML consente di descrivere un sistema secondo tre aspetti principali:
  - il modello **funzionale** (diagrammi dei casi d'uso)
  - il modello **a oggetti** (diagrammi delle classi)
  - il modello **dinamico** (diagrammi di sequenza)

- La documentazione del progetto è uno strumento che obbliga ad una attenta riflessione utile per evitare di compiere errori nella successiva fase di realizzazione
- UML offre uno strumento di documentazione vicino e compatibile con gli aspetti più tecnici di realizzazione del software; adotta inoltre un formalismo noto e comune; per questi due motivi principali ha avuto un successo notevole
- UML viene usato per descrivere e documentare aspetti assai complessi ed astratti, per cui non è di per sé sufficiente: va integrato con documentazione più descrittiva o di maggiore dettaglio su alcuni aspetti

UML, modello funzionale: use case diagrams

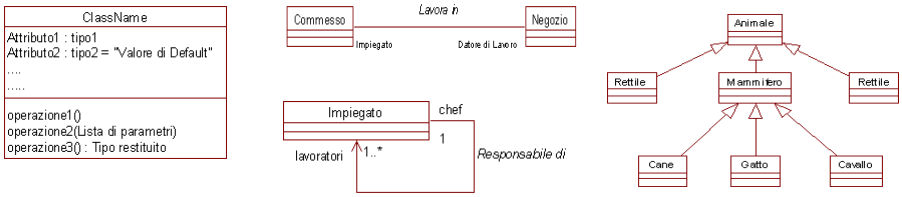
- Il **caso d'uso** (*use case*) è la descrizione di uno scenario elementare di utilizzo del sistema
- La specifica di un caso d'uso dovrebbe includere
  - un **nome**, con cui identificare il caso d'uso
  - gli **attori** principali e secondari, ossia i soggetti (umani o tecnologici) che partecipano allo scenario
  - un **obiettivo**, il motivo per cui gli attori principali avviano il caso d'uso
  - la **precondizione** nella quale è eseguibile il caso d'uso
- Lo ***use case diagram*** di UML fornisce una rappresentazione del caso d'uso



UML, modello a oggetti: class diagrams

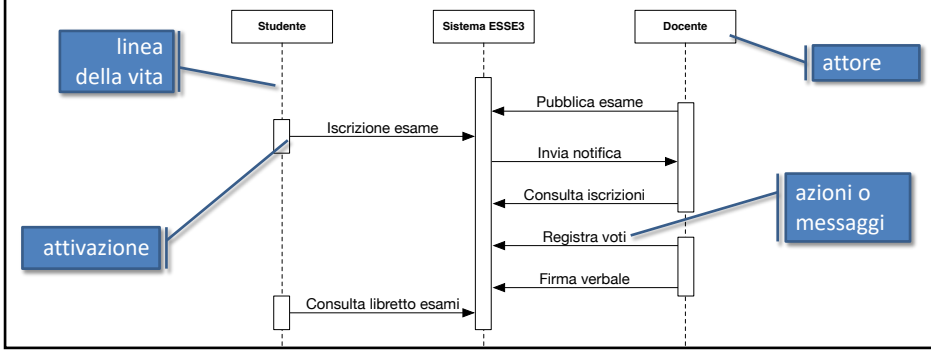
- Nella programmazione *object oriented* la "**classe**" è il concetto fondamentale su cui si articola l'intero programma: descrivere correttamente le classi con i loro metodi e attributi e le relazioni esistenti tra oggetti di classi diverse è di cruciale importanza
- I ***class diagram*** forniscono una **visione statica** delle classi presenti in un progetto software object oriented

- Il *class diagram* del linguaggio UML rappresenta le classi come box contenenti gli attributi della classe e collegate fra loro da linee che rappresentano i diversi tipi di relazione esistenti tra le classi
  - la **molteplicità** è un tipo di associazione in cui si mostra il numero di oggetti appartenenti ad una classe che interagisce con il numero di oggetti della classe associata
  - l'**ereditarietà** mostra le proprietà che vengono condivise tra classe padre e classe figlio



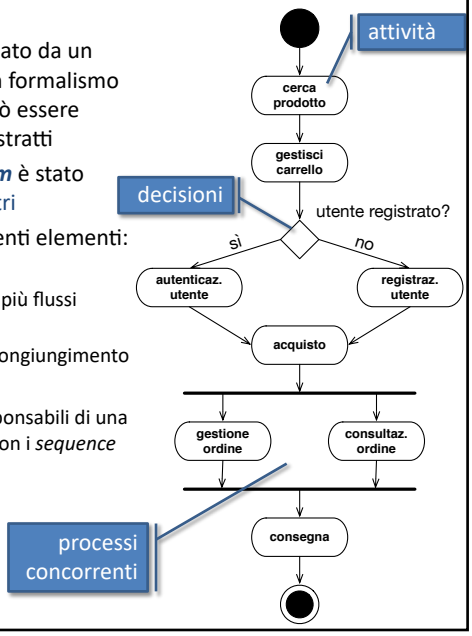
UML, modello dinamico: sequence diagram

- Un **diagramma di sequenza** (*sequence diagram*) descrive uno scenario, evidenziando la **sequenza** di azioni svolte dagli attori sul sistema (o messaggi scambiati tra gli attori e il sistema)
- Uno **scenario** è una determinata sequenza di azioni in cui tutte le scelte sono state già effettuate; in pratica nel diagramma non compaiono scelte, né flussi alternativi
- Normalmente da ogni *activity diagram* sono derivati uno o più *sequence diagram*
  - ad esempio se un activity diagram descrive due flussi di azioni alternativi, se ne possono ricavare due scenari, e quindi due sequence diagram alternativi



UML, modello dinamico: activity diagram

- Rappresenta i passi di un processo implementato da un sistema per realizzare le sue funzionalità; è un formalismo simile a quello dei diagrammi di flusso, ma può essere impiegato in contesti di minor dettaglio, più astratti
- In UML 2.x il formalismo degli ***activity diagram*** è stato ridisegnato per seguire quello delle Reti di Petri
- In un *activity diagram* sono evidenziati i seguenti elementi:
  - azioni
  - decisioni (che portano a dividere il processo in più flussi distinti)
  - punti di separazione del processo o punti di ricongiungimento
  - punti di avvio e di conclusione del processo
  - possono essere anche evidenziati gli attori responsabili di una determinata azione (creando una ibridazione con i *sequence diagram*)



Altri diagrammi

- *Business Process Model Notation* (BPMN)
  - rappresentazione e descrizione di un processo di business (workflow)
- *Data Flow Diagram*
  - rappresentazione del flusso delle informazioni tra i diversi moduli di un sistema informatico che si scambiano dati attraverso aree di memoria comuni, archivi condivisi, protocolli di rete o meccanismi di interprocess communication
- *State Diagram*
  - rappresenta gli stati in cui può trovarsi un sistema e le transizioni di stato lecite (o gestite/implementate dal sistema stesso) che consentono il passaggio da uno stato ad un altro

Gestione della configurazione

- Il **configuration management** supporta la gestione ed il controllo degli elementi che compongono un sistema software
- La gestione di tali elementi (***configuration items***) è affidata ad un database (**repository** di configurazione) in cui sono censiti gli oggetti sottoposti a controllo di configurazione
- Nell'ambito del ***configuration management*** vengono gestiti gli input/output direttamente o indirettamente legati alla costruzione di un prodotto software
  - Si archiviano in modo controllato le varie versioni del codice sorgente sviluppato, i documenti, i test, i dati di configurazione, ecc.
- Una delle funzioni svolte da un **sistema di controllo della configurazione** (**CMS**, *configuration management system*) è quella di correlare tra loro i vari oggetti archiviati relativamente alle diverse ***versioni*** o ai diversi ***branch*** di un prodotto software
- **Versione**: è un sottoinsieme dei *configuration item* che definisce completamente una release del prodotto software;
  - possono coesistere più versioni successive dello stesso prodotto (manutenzione correttiva ed evolutiva di versioni precedenti a quella corrente, con modifiche che sono recepite dalle versioni successive)
  - uno stesso configuration item può essere presente in una o più versioni del prodotto
- **Branch**: è una copia di una versione del prodotto, in modo da portare avanti versioni diverse e indipendenti del prodotto (le modifiche su un *branch* non impattano sugli altri)

Gestione della configurazione

- Un prodotto CMS è un software che gestisce un archivio di file e offre un protocollo di comunicazione per eseguire delle operazioni sul **repository** del prodotto software
  - ***check out***: si estraggono dal *repository* i *configuration item* e se ne produce una copia locale su cui l'utente (programmatore) può lavorare
  - ***update***: si aggiorna il progetto locale con eventuali modifiche occorse sui *configuration item* presenti sul *repository* di progetto
  - ***add***, ***delete***, ***move***, ***copy***: si aggiunge o si modifica la configurazione del progetto con elementi presenti nella configurazione locale (nuovi file, eliminazione di file, ecc.)
  - ***commit***: si carica sul *repository* la copia locale del progetto (o di singoli *configuration item*)
  - ***lock***: si blocca sul repository la modifica ad un *configuration item* che è in lavorazione da parte di un utente (programmatore) sul proprio *repository* locale
- Il software CMS rileva eventuali **conflitti** (es.: aggiornamenti diversi da parte di due utenti allo stesso *configuration item*) e aiuta gli utenti del gruppo di lavoro a gestirli
- Il software CMS ad ogni aggiornamento di un *configuration item* ne crea una **nuova versione**, mantenendo in archivio le vecchie versioni (*versioning*)
- Il software CMS gestisce il **controllo degli accessi degli utenti** ai repository dei diversi prodotti software

Gestione della configurazione

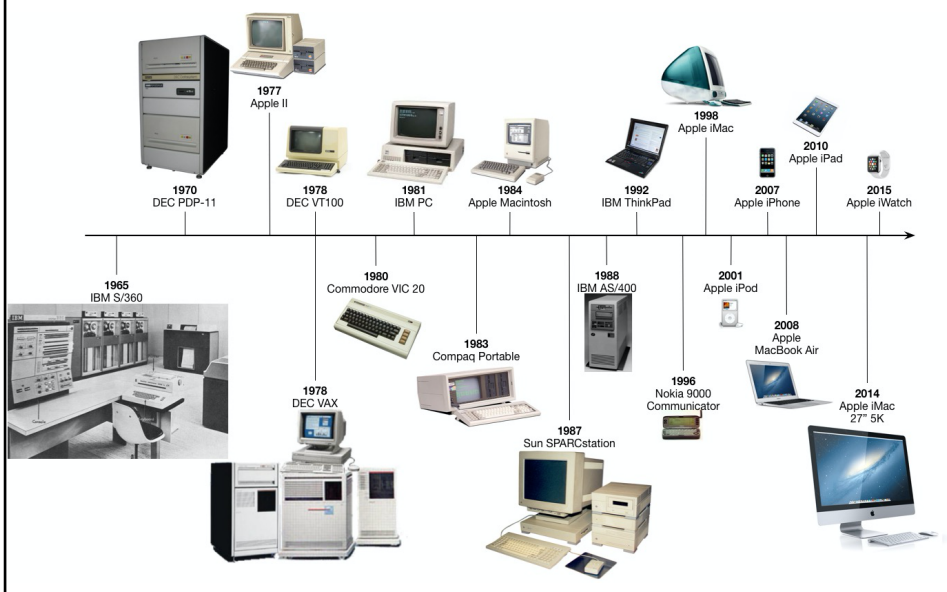
- Alcuni prodotti software di configuration management
  - **CVS**, *concurrent version system* (open source)
  - **SVN**, *subversions*, nato come evoluzione di CVS (open source)
  - **GIT**, realizzato da Linus Torvalds per il controllo di configurazione del kernel di Linux (open source)
  - **IBM Rational ClearCase**, componente della suite di tool di progettazione e sviluppo software IBM Rational (proprietario)
  - **Microsoft Visual SourceSafe**, componente della suite di tool di sviluppo software Microsoft Visual Studio (proprietario)

Sistemi di Gestione per la Qualità

- Terminologia:
  - **Prodotto**: **output** di un processo di lavorazione
  - **Processo di lavorazione:** un **progetto** per realizzare un sistema informatico, un insieme di attività per realizzare un **prodotto**, un insieme di attività per erogare un **servizio**, ecc.
  - **Requisiti**: **caratteristiche** che ci si aspetta debba avere il prodotto
  - **Prodotto di Qualità: Prodotto** che rispetta in pieno ai **requisiti**
- **Sistemi di Gestione per la Qualità**
  - Complesso di **procedure**, **regole** e **strumenti** (tecnici e documentali) di cui un'organizzazione si dota per gestire il processo produttivo in modo tale da realizzare prodotti di qualità
  - Si basano su alcuni principi fondamentali:
    - DOCUMENTAZIONE: descrizione esplicita delle informazioni necessarie per la realizzazione del prodotto
    - TRACCIABILITÀ: identificare con certezza lo stato del processo, delle attività e delle componenti in lavorazione
    - IDENTIFICAZIONE: possibilità di identificare con certezza ogni elemento che entra a far parte del processo produttivo, rendendo evidenti le modifiche e lo stato di aggiornamento di ciascun elemento
    - RIPRODUCIBILITÀ: possibilità di ripetere un processo (è possibile se è documentato)
    - MISURAZIONE: ogni attività deve essere caratterizzata da un insieme di indicatori che consentano di "misurare la qualità"
- Normativa internazionale che definisce i requisiti di un *buon* sistema di gestione per la qualità: **UNI EN ISO 9001:2000** (ultima revisione: novembre 2008)

Sistemi di Gestione per la Qualità

- Un Sistema di Gestione per la Qualità certificato secondo la norma ISO 9001 è caratterizzato da un insieme di documenti:
  - **Politica per la qualità**: documento con cui il top management dell'azienda dichiara la motivazione ad adottare il sistema e le linee guida di alto livello per la definizione del sistema
  - **Organigramma funzionale e nominale**: è il documento in cui si dichiara la struttura organizzativa e i ruoli
  - **Manuale della qualità**: documento che descrive macroscopicamente i processi presenti nell'organizzazione e i workflow che determinano i contributi delle diverse figure professionali
  - **Procedure gestionali**: ogni processo macroscopico viene descritto in una procedura che evidenzia gli step e le responsabilità e indica in che modo il processo viene documentato (es.: processo selezione del personale, processo acquisto, processo progettazione e sviluppo software, ecc.)
  - **Istruzioni operative**: servono per descrivere in dettaglio specifiche operazioni svolte nell'ambito di un processo (es.: gestione della corrispondenza in entrata o in uscita, creazione di un nuovo ambiente di sviluppo per un progetto software, ecc.)
  - **Modulistica**: sono documenti utili per le "registrazioni della qualità" e per la rilevazione degli indicatori di performance
- Il sistema di gestione per la Qualità si applica ad ogni progetto di sviluppo software
- Per progetti particolarmente rilevanti è possibile definire un **Piano della Qualità** specifico per un determinato progetto
- È importante definire una **codifica** univoca per **identificare** i documenti prodotti nell'ambito del progetto



1965 – 2015: alcune milestone degli ultimi 50 anni di informatica
Alcuni computer che negli ultimi 50 anni hanno segnato dei passi in avanti significativi nell'evoluzione della tecnologia informatica