

Per effettuare correttamente l'analisi è necessario comunicare con l'utente, ottenere buone conoscenze sull'area applicativa e determinare in dettaglio i requisiti di sistema.

### **Analisi e gestione dei rischi:**

Si utilizza una strategia preventiva, cioè individua i rischi potenziali valutandone la probabilità e gli effetti scatenati e si studia un piano per reagire in modo adeguato. Possono esistere rischi relativi ai requisiti, alle risorse umane, rischi tecnologici e politici.

### **Raccolta dei requisiti:**

La parte più importante del progetto. Si parla col cliente e si cerca di reperire tutte le informazioni su cosa il sistema deve fare secondo le intenzioni del cliente. E' necessario chiedere al cliente quali sono le attività più affaticanti, più preoccupanti e qual è la cosa peggiore che potrebbe capitare durante il lavoro degli utenti.

La raccolta dei requisiti deve produrre un documento con tutti i requisiti scritto dall'analista e approvato dal cliente e un dizionario con la definizione di TUTTI i termini e i concetti utilizzati.

Ogni requisito deve essere validato e negoziato con i clienti.

Bisogna tenere in conto che i cambiamenti dei requisiti sono la norma e non l'eccezione, e potrebbe essere un grande problema non riuscire a gestirli opportunamente; in particolare più si va avanti nel progetto, più è difficile e costoso modificare dei requisiti. Si rende quindi necessario sviluppare dei sistemi che siano resistenti ai cambiamenti, e che possano sopportare l'aggiunta di funzionalità senza far crashare il sistema o senza farlo comportare in modo anomalo.

### **Analisi del dominio:**

Creare un modello che descriva il dominio del problema. Bisogna trovare le strutture e i comportamenti comuni a tutti i sistemi di una certa area applicativa. Lo scopo principale è quello di permettere la riusabilità degli schemi di progettazione.

### **Analisi dei requisiti:**

Il modello descrive le responsabilità del sistema. Bisogna definire i requisiti funzionali e descrivere il comportamento del sistema.

Dall'analisi dei requisiti deve uscire un documento dei requisiti in cui sono spiegate le necessità del cliente e gli obblighi del progettista.

### **Analisi di casi d'uso e scenari:**

Tramite questi è possibile formalizzare i requisiti funzionali e permettere di capire meglio il funzionamento del sistema.

Caso d'uso: descrizione di una richiesta fatta al sistema da parte di un'entità esterna. L'insieme dei casi d'uso è l'immagine del sistema verso l'esterno.

Bisogna individuare il confine del sistema, gli attori e i casi d'uso.

Un caso d'uso viene sempre avviato direttamente o indirettamente da un attore che ha un obiettivo, e si conclude con il successo o l'insuccesso dell'obiettivo.

Un caso d'uso è sempre descritto dal punto di vista dell'attore ed ha delle precondizioni (facoltative), una sequenza di passi per ottenere l'obiettivo, e delle postcondizioni (facoltative). La sequenza di passi deve utilizzare i termini del

vocabolario del dominio: se alcuni termini importanti non ci sono, vanno aggiunti nel glossario.

Scenario: descrizione della sequenza di passi che descrive l'interazione tra attore e sistema e le elaborazioni per soddisfare la richiesta dell'utente.

Tipicamente per ogni caso d'uso c'è uno scenario principale ed eventuali scenari alternativi.

Ogni scenario viene descritto mediante linguaggio naturale perché sia ben comprensibile dall'utente che così si sentirà più coinvolto nell'analisi e contribuirà a specificare meglio gli scenari.

Bisogna identificare tutti i possibili utilizzi del sistema, definire un attore per ogni categoria di utenza e per ogni ruolo, identificare tutti gli obiettivi di tutti gli utenti e per ognuno di essi definire un caso d'uso. I casi d'uso non devono eccedere nella complessità e nei dettagli e devono sempre mantenere lo STESSO livello di astrazione.

### **Modello statico:**

Si tratta di un approccio orientato agli oggetti che segue i principi dell'incapsulamento (i dati devono mostrare il meno possibile delle proprie caratteristiche interne) e dell'ereditarietà (attributi comuni specificati una sola volta).

Si usa la notazione UML in cui ogni classe è graficamente divisa in tre sezioni: stereotipo e nome della classe, attributi, operazioni.

Individuazione delle classi: utilizzando il documento dei requisiti e riutilizzano le stesse strutture emerse da precedenti analisi dello stesso dominio. Il nome della classe deve essere un nome familiare singolare e iniziare con la lettera maiuscola. Bisogna individuare gli attori, le cose tangibili, i contenitori e gli eventi del sistema. Attenzione agli oggetti che hanno un solo attributo, alle classi che hanno un solo oggetto, oggetti diversi con le stesse responsabilità.

Individuazione delle responsabilità: una responsabilità è la descrizione ad alto livello di quello che una classe deve fare. Bisogna cercare i verbi nel documento dei requisiti che rappresentano le azioni fatte dagli oggetti. Le responsabilità devono essere distribuite uniformemente tra le classi (non deve esserci una classe con 10 responsabilità diverse e una con nessuna), ed essere di astrazione inversamente proporzionale alla specificità della classe.

Individuazione delle relazioni: cioè come una classe è connessa ad un'altra (ereditarietà, dipendenza, associazione, istanza-classe, classe-metaclasse).

Individuazione dell'ereditarietà: una classe eredita da un'altra se deriva da una modifica dell'altra classe, cioè se è una sua specializzazione e ne eredita relazioni, attributi e operazioni. L'ereditarietà può essere semplice (una classe ha una sola superclasse) o multipla (ha più superclassi); quest'ultima, ammessa in fase di analisi, deve essere eliminata in fase di progettazione.

Individuazione delle collaborazioni: una classe "usa" un'altra se ha bisogno della collaborazione dell'altra classe per svolgere alcune funzionalità.

Individuazione delle associazioni: un'associazione può rappresentare il contenimento logico (aggregazione) di un'altra classe se l'associazione è non esclusiva e la cancellazione fisica del contenitore non comporta la cancellazione del contenuto; il contenimento fisico (composizione) comporta un contenimento esclusivo e la distruzione del contenitore comporta la distruzione del contenuto. Nel caso di contenimento i due oggetti devono avere un legame forte e lo stesso ciclo di vita. Le associazioni hanno delle cardinalità distinte da un limite inferiore (connessione facoltativa o obbligatoria?) e un limite superiore (connessione singola

o multipla?). Nel caso di connessioni molti a molti possono esserci delle classi associazioni; in questo caso le due classi hanno un legame 1 a 1 con la classe associazione. La classe associazione può essere concretizzata nel caso in cui sia necessario definire un diverso legame con la classe associazione.

Individuazione degli attributi: ogni attributo modella una proprietà atomica di una classe. Deve avere un nome familiare per l'utente, iniziare con la lettera minuscola, non deve rappresentare il "nome" di un valore (che quindi presupporrebbe un attributo di tipo si/no) o il "tipo" ed essere di tipo "private". Nel caso di attributi calcolabili, si tratta di un'operazione. Attenzione ai casi in cui una classe non abbia attributi né associazioni: va eliminata. Nel caso di ereditarietà, gli attributi e le operazioni più generali vanno posizionate più in alto di quelli specializzati.

Individuazione delle operazioni: le operazioni sono in stretto rapporto con le responsabilità della classe. Devono avere un nome familiare all'utente, potrebbe essere un verbo (alla 3 persona singolare o all'imperativo) e iniziare con la lettera maiuscola (convenzione .NET). Possono avere una lista di parametri e un valore restituito. In fase di analisi non si menzionano le get e le set di ogni attributo, né tutte quelle operazioni di creazione, memorizzazione, lettura dell'oggetto che verranno eventualmente riportate in fase di progettazione. Possono essere metodi astratti (in corsivo) o di classe (sottolineato). Come per gli attributi, nel caso di ereditarietà le operazioni più astratte devono essere più in alto di quelle più specifiche. Le operazioni possono avere precondizioni, postcondizioni e invarianti di classe.

Le precondizioni sono espressioni logiche che riguardano le aspettative sullo stato del sistema prima dell'inizio dell'operazione; le postcondizioni riguardano le aspettative sullo stato del sistema al termine dell'operazione; gli invarianti di classe sono vincoli che devono essere verificati sempre all'inizio e alla fine di tutte le operazioni pubbliche della classe. In caso di ridefinizione di un'operazione in una sottoclasse le precondizioni possono essere identiche o meno stringenti e le postcondizioni possono essere identiche o più stringenti.

Un'eccezione si verifica quando un'operazione, invocata rispettando le precondizioni, non riesce a terminare nel rispetto delle post-condizioni.

### **Modello dinamico:**

Fornisce il comportamento nel tempo del sistema.

Partendo dai casi d'uso, l'invio di un messaggio deve corrispondere ad un'operazione della classe del destinatario o della superclasse.

Diagrammi di sequenza: evidenzia l'ordine con cui i messaggi vengono scambiati.

Per ogni oggetto bisogna chiedersi se si è in grado di soddisfare la richiesta in modo autonomo, se non è possibile ci si chiede a quale altro oggetto bisogna riferirsi per ottenere i dati mancanti e si itera il procedimento fino all'identificazione di tutta la sequenza dei messaggi e al soddisfacimento della richiesta.

NB: un oggetto può mandare un messaggio ad un fornitore solo se conosce il fornitore e attraverso l'esecuzione di un proprio metodo.

Diagrammi di collaborazione: evidenzia la connessione tra oggetti e messaggi scambiati. Una connessione tra oggetti mostra l'accesso di un oggetto ad un altro oggetto.

Diagrammi di stato: mostra i possibili stati che può raggiungere un oggetto in base alla ricezione di determinati eventi.