**04/02** - Tundo, Pigazzini, Popescu

definizione attori che interagiscono con il sistema: discussione riguardo cliente e outsourcer, due possibilità:

* outsourcer considerato attore, con una propria vista nel sistema
* solo notifiche, ad esempio al cliente verrebbe inviata una mail per aggiornarlo sull'avanzamento dei progetto o per i meeting.

Problema fondamentale: Gestione dei ruoli/permessi nei differenti progetti

All'atto della creazione di un progetto il Project Manager (PM) stabilisce le risorse da allocare per il progetto, chi sono i responsabili (R) delle varie fasi e gli sviluppatori (DEV). I R possono a loro volta decidere i ruoli dei vari sviluppatori.

Chi decide però chi è un PM? Come fa ad avere dei diritti diversi da un semplice DEV dato che il controllo sul ruolo/permessi viene fatto quando visualizzo (entro in) un progetto?

Assumiamo che chi crea il progetto è il suo PM, e per poterlo creare l'utente deve essere in possesso di una chiave univoca fornita fisicamente dal responsabile IT / capo. Questa parte di generazione della chiave univoca non verrà modellata in quanto non è obiettivo principale del progetto. Si supporrà quindi che quando nel sistema si avvierà la creazione di un nuovo progetto verrà richiesta la chiave e da quel momento in poi quell'utente diverrà PM per quel progetto.

**05/02** Tundo, Pigazzini

* Caso d’uso dettagliato Create Project
* *Problema*: una persona può essere implicata contemporaneamente nello sviluppo di due o più progetti.  
  *Soluzione*: Ad una persona possono essere assegnati più task ma non sovrapposti nel tempo (tempoInizioTask2 > tempoFineTask1).  
  Inoltre per poter effettuare i calcoli sulle durate, la gestione dei ritardi ecc si assume che una persona lavora 8 ore al giorno per 5 giorni a settimana (40 ore settimanali), quindi si usa come unità di tempo l’ora.  
  Se la persona è responsabile di una fase, può tranquillamente esserlo di un’altra ma vale la stessa regola precedentemente descritta (tempoInizioFase2 > tempoFineFase1).  
  Allo stesso modo se una persona è sviluppatore di un task, può anche essere responsabile di una fase nello stesso o in un altro progetto, solo che queste non possono sovrapporsi temporalmente.
* Si è deciso di categorizzare gli utenti. Ora PM, Senior e Junior sono sottoclassi della classe Utente definite in questo modo: un utente PM può assumere ruolo di PM per un progetto, e ruolo di Supervisor per uno stage; un utente Senior può assumere ruolo di Supervisor per uno stage e ruolo di Developer per un task; un utente Junior può assumere solamente ruolo di Developer per un task.

**08/02** Tundo, Pigazzini

* Definizione dei commenti brevi ai singoli casi d’uso. Inizio definizione dettagliata del caso d’uso ManageProject
* I task possono essere sovrapposti e questo è possibile grazie al calcolo descritto più avanti.
* Raffinamento Domain Model
* **Proposta calcolo disponibilità risorsa umana**:
  + il supervisor crea i task, con una certa durata (data inizio – data fine)
  + decide la quantità di persone per portare a termine il task
  + il sistema calcola il numero di ore richiesto per persona per il task in considerazione (es.: *oreRichiesteAPersona = numeroGiorniLavorativi\*8 / numeroPersone*)
  + il sistema calcola se una certa persona è disponibile:

*totaleOreLavorativeCoperturaTasks = 8\*sommaGiorniLavorativiTasks*

*oreDisponibiliPeriodo = (totaleOreLavorativeCopertaTasks – oreRichiesteAltriTasks*)

se *oreDisponibiliPeriodo >= oreRichiesteAPersona* allora posso assegnare quel task alla persona in considerazione.

* **Proposta calcolo completamento task:** booleano nel task e classe associativa che tiene traccia di chi ha completato la sua parte di task. Quando tutti gli sviluppatori di un task hanno settato il booleano della classe associativa, allora il booleano del task viene settato a TRUE.

Un task quindi è o completo al 100% o non completo.

* **Proposta calcolo percentuale completamento stage:**

Ogni task che compone il progetto ha un'influenza differente sulla percentuale totale a seconda del rapporto fra la sua durata e quella totale dello stage.

*PercentualeCompletamentoStage = SommaPercentualiTaskCompletati*

Esempio: durataStage(10 giorni lavorativi), composto da 3 task, rispettivamente di 5, 3 e 2 giorni lavorativi.

Quando, per esempio, il primo task viene settato come completo la percentuale di completamento dello stage diventerà 50% e così via...

* **Proposta calcolo percentuale completamento progetto:**

Stessa idea del calcolo completamento stage (task → stage, stage → project).

* Anche il ruolo di supervisor e pm “consuma” il tempo giornaliero disponibile. Sarebbe opportuno poter impostare da un file di configurazione quante ore sottrarre per le seguenti attività:
  + project manager (ad esempio 4 ore al giorno per progetto)
  + supervisor (ad esempio 2 ore al giorno per stage)

**09/02** – Nava, Pigazzini, Tundo

* Correzione del **calcolo disponibilità risorse umane**:
* *oreDisponibiliPeriodo = [(totaleOreLavorativeCopertaTasks – oreRichiesteAltriTasks*) *– 8\*differenzaDateInizioTasks]*
* Dobbiamo usare ASSOLUTAMENTE un Template Engine per le viste (JSP, Freemaker, Mustache, Thymeleaf, Jade4j)
* Per ripetere una determinata operazione giornaliera (es. controllo stato progetti ed eventuali ritardi) si può utilizzare Java Scheduled Executor Service.
* Raffinamento Domain Model
* **Calcolo parametri CPM**
  + Il PM inserisce la durata(stimata) del progetto e la data di inizio
  + Al momento della creazione delle fasi da parte del PM viene richiesta la durata di una fase (in giorni)
  + Per ogni fase si indicano le precedenze
  + Grazie a queste due informazioni si calcolano ES, EF, LS, LF, ottenendo così anche le date di fine e le attività critiche
  + Il sistema potrebbe già in questa fase avvertire il PM che la deadline da lui precedentemente inserita non può essere rispettata
  + Tutti i giorni (di notte), il sistema avvia una procedura schedulata che si occupa controllare se una fase non ha rispettato la sua deadline. In tal caso, se quest'ultima è critica effettua tutti i calcoli del caso(es. quanto slitta la data di fine del progetto) e notifica un ritardo critico al PM ed al responsabile di quella fase.
* Proposta requisito **SEARCH:**

Un progetto possiede un insieme di “aree di interesse” tramite le quali viene catalogato. In questo modo posso risalire grazie a questi “metadati” ai clienti che hanno fatto richiesta di progetti simili ed anche i progetti stessi. Quindi ad esempio:

* + Ricerca di clienti: inserisco delle aree di interesse e il sistema restituisce tutti i clienti con i relativi progetti che fanno parte di quell'area.
  + Ricerca di progetti: inserisco delle aree di interesse ed il sistema restituisce tutti i progetti simili.

**10/02** - Vazzola, Tundo, Pigazzini

* Raffinamento Domain Model
* Raffinamento Use Case Diagram
* Scrittura casi d'uso informali(manca Project Tracking perché probabilmente verrà cambiato o comunque diviso in più casi d'uso)
* Raffinamento Create Project Use Case
* Create Project SSD
* Inizio Caso d'uso dettagliato Manage Stage (mancano estensioni, in particolare controllare l'assegnazione della risorse umane!!!!!)

**11/02** – Vazzola, Tundo, Popescu, Pigazzini, Nava

* Raffinamento Use Case Diagram
* Discussione gestione ritardi, assegnazione risorse umane, ottimizzazione risorse
* Continuazione ManageStage

**12/02** – Pigazzini, Tundo

* Creazione diagramma SSD di “SearchProjectUseCase”
* TO DO diagramma di sequenza di “ManageDelaysUseCase” (da fare dopo che si è definito il diagramma delle classi di design)

**16/02** – Vazzola, Pigazzini, Tundo, Popescu

* Inizio implementazione codice Java
* Creata query di restituzione di tutti i PM coinvolti in vari progetti

**18/02** – Vazzola, Pigazzini, Tundo, Popescu, Nava

* Modellazione delle classi su Modelio
* Creazione dei diagrammi delle classi e diagramma di attività
* Refactoring tramite Sonarqube

**19/02** - Vazzola, Pigazzini, Tundo, Popescu, Nava

* Prosecuzione implementazione codice Java nel caso non ci siano PM disponibili e quindi opzione di fare outsourcing.
* Creazione di un possibile algoritmo per la classe “calcola\_ore” che consente di capire quante ore sono disponibili ancora a ogni persona per poter assegnare eventualmente altri task.

**22/02** - Vazzola, Pigazzini, Tundo, Popescu, Nava

* Raffinamento della classe “calcola\_ore” e crezione della classe di test associata
* Discussione generale e abbozzo di un possibile algoritmo per la gestione di eventuali ritardi al progetto(Es,Ef,Ls,Lf).

**23/02** - Vazzola, Pigazzini, Tundo, Popescu, Nava

* Aggiornamento diagramma delle classi su Modelio
* Discussione generale sui meeting e bozza della pagina di visualizzazione per l’organizzazione del meeting
* Implementazione in Java delle classi “CalculateWeights” e “UpdateRateCompleted”

**24/02** - Vazzola, Pigazzini, Tundo, Popescu, Nava

* Aggiornamento diagramma delle classi su Modelio
* Implementazione della servlet “OrganizeMeetingServlet” e della relativa pagina jsp
* Implementazione del metodo per calcolare il percorso critico

**26/02** - Vazzola, Pigazzini, Tundo, Popescu

* Aggiornamento diagramma delle classi su Modelio
* Modificato StageDAO (getStagesByIdProject)
* Eseguito refactoring su security (applicato Strategy)
* Eseguite correzioni nella query dei developers e altre modifiche minori