Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Analisi e confronto dei giochi di team building per sviluppi Agili

Relatore:
Chiar.mo Prof.
PAOLO CIANCARINI

Presentata da:
FRANCESCA CHIRIACÒ

Sessione III

Anno Accademico 2023-2024

Sommario

I Serious Games nascono da una fusione innovativa tra il gioco, visto come una pura attività ludica, e l'educazione e formazione professionale con lo scopo di conseguire obiettivi legati all'apprendimento o allo sviluppo di competenze specifiche. Attraverso la simulazione di scenari, sfide da superare e compiti da portare a termine, i Serious Games creano un ambiente che stimola i partecipanti a pensare, agire e collaborare in modi che riflettono situazioni reali.
Grazie alla loro capacità di coinvolgere i giocatori in esperienze pratiche e significative, questi giochi rappresentano strumenti efficaci nel promuovere l'apprendimento dei valori Agili e nel facilitare il processo di team building all'interno di un gruppo. Attraverso i Serious Games, i partecipanti imparano a valorizzare le diverse competenze e prospettive all'interno del team, cooperando per raggiungere obiettivi comuni. Questo tipo di interazione, oltre a rafforzare le relazioni interpersonali, crea un senso di comunità e fiducia reciproca, concetti alla base di ogni team Agile.
Questa tesi si propone di dimostrare come l'integrazione di questi strumenti ludici nel percorso formativo possa non solo migliorare la comprensione dei principi Agili, ma anche accelerare il processo di trasformazione necessario per il successo di un team di sviluppo software in un ambiente lavorativo in rapida evoluzione.

Indice

1 Introduzione **1**
- 1.1 Sviluppo Agile . . . 1
- 1.2 Team building . . . 4
  - 1.2.1 Cooperative Thinking . . . 5
- 1.3 Team Coaching . . . 7
- 1.4 Serious Games . . . 8

2 Scrum **13**
- 2.1 Framework Scrum . . . 13
  - 2.1.1 Product Owner . . . 14
  - 2.1.2 Scrum Master . . . 16
  - 2.1.3 Team di sviluppo . . . 17
  - 2.1.4 Artefatti Scrum . . . 18
  - 2.1.5 User Story . . . 20
  - 2.1.6 Sprint . . . 22
  - 2.1.7 Pianificazione dello sprint . . . 22
  - 2.1.8 Retrospettive . . . 23

3 Giochi per la formazione del Product Owner **24**
- 3.1 Escape the Boom . . . 24
  - 3.1.1 Adattamento del gioco . . . 25
  - 3.1.2 Fasi di Escape the Boom . . . 25
- 3.2 Scrumble . . . 26
  - 3.2.1 Fasi di Scrumble . . . 27
- 3.3 Product Owner Value Game . . . 29
  - 3.3.1 Fasi di Product Owner Value Game . . . 29

4 Analisi e confronto dei giochi proposti **32**
- 4.1 Approccio GQM . . . 32
  - 4.1.1 Valutazione GQM di una partita a Escape the Boom . . . 34
  - 4.1.2 Valutazione GQM di una partita Scrumble . . . 35

4.1.3 Valutazione GQM di una partita a Product Owner Value Game . 37

5 Giochi di retrospettiva con Essence **39**
5.1 Framework Essence 39
5.2 Carte Essence 42
5.3 Retrospettive con carte Essence 42
5.3.1 Team Status Game 43
5.3.2 Alpha State Cards Game 44
5.3.3 Scrum 45

6 Conclusioni **47**

Elenco delle figure

1.1 Fasi dello sviluppo Agile [43] . . . 3
1.2 Fasi di sviluppo di un team secondo Tuckman [22] . . . 5
1.3 Il pensiero computazionale ed i valori agili di collaborazione sono costrutti componenti del Cooperative Thinking [9] . . . 7
1.4 Serious Games [39] . . . 9
1.5 Sprint in Yata [31] . . . 9
1.6 Carte Pipeline [3] . . . 11
1.7 Carte Lego City [32] . . . 12

2.1 Framework Scrum [33] . . . 14
2.2 Attività principali del Product Owner [42] . . . 15
2.3 Triangolo di Ferro [24] . . . 16
2.4 Attività dello Scrum Master [42] . . . 17
2.5 Attività del Team di sviluppo [42] . . . 18
2.6 Grafico Burn Down [30] . . . 19
2.7 Template User Story [12] . . . 20
2.8 Modello PRIUS [26] . . . 22

3.1 Escape the Boom [13] . . . 25
3.2 Tabellone Scrumble [37] . . . 27
3.3 Fattori che producono debito tecnico [37] . . . 28
3.4 Carte del Product Owner Value Game [15] . . . 29
3.5 Carte Feature [15] . . . 30
3.6 Carte User Stories [15] . . . 31

4.1 Struttura a livelli del metodo GQM [29] . . . 33
4.2 Tabella GQM per la valutazione di Escape the Boom . . . 35
4.3 Tabella GQM per la valutazione di Scrumble [6] . . . 36
4.4 Tabella GQM per la valutazione di Product Owner Value Game . . . 37

5.1 Framework Essence: i sette alpha principali e le loro relazioni [19] . . . 40
5.2 Alpha State Cards Game . . . 44

5.3 Retrospettiva con carte Essence 45

Capitolo 1

Introduzione

In questo capitolo introduttivo si esplorerà come le attività di team building possano rappresentare un approccio dinamico e innovativo per lo sviluppo delle competenze collaborative all'interno di un team Agile. In seguito, l'attenzione sarà posta sul Cooperative Thinking, sull'attività di team coaching e sull'uso di Serious Games, esaminando come la concatenazione di queste metodologie possa trasformare in maniera radicale il modo in cui i team di sviluppo software collaborano ed evolvono.
Il Cooperative Thinking rappresenta un cambio di paradigma nell'approccio al lavoro di squadra,in quanto pone maggior interesse sulla cooperazione anziché sulla competizione. Quindi, vedremo come questa nuova prospettiva possa ispirare il pensiero creativo, favorendo l'innovazione e la coesione di gruppo.
Il team coaching, d'altra parte, si presenta come uno strumento essenziale per guidare i team nel loro percorso di crescita. Verranno, quindi, analizzate le tecniche e le strategie adottate dai coach per sviluppare le capacità di leadership, la comunicazione efficace e la risoluzione dei conflitti all'interno dei team.
Infine, sarà introdotto il mondo dei Serious Games, un approccio ludico volto a stimolare l'apprendimento e lo sviluppo delle competenze di squadra. Si analizzerà come questi giochi possano trasformare l'apprendimento in un'esperienza coinvolgente e divertente.

1.1 Sviluppo Agile

La metodologia di sviluppo Agile nasce formalmente nel 2001, quando 17 specialisti sviluppatori software si riuniscono nello Utah pubblicando il Manifesto Agile [4]. Tale documento introdusse i seguenti quattro valori utili ai professionisti per approfondire le basi di una buona gestione dello sviluppo software:

1. **Preferiamo gli individui e le interazioni più che i processi e gli strumenti:** individui e interazioni vengono preferiti in quanto rendono il team più efficace ed efficiente. Sono i membri del team ad avere la responsabilità di decidere cosa sia

necessario fare, di scoprire cosa li rallenta nell'esecuzione e di portare a termine i vari compiti. Se i membri di un team collaborano e si aiutano tra loro, il gruppo può risolvere i vari problemi derivanti dallo sviluppo trovando rapidamente le soluzioni più adeguate.

2. **Preferiamo il software funzionante più che la documentazione esaustiva:** solitamente, un progetto di sviluppo prevede una folta e dettagliata documentazione (specifiche tecniche, requisiti tecnici, documenti di progettazione dell'interfaccia..) la cui preparazione, molto spesso, porta ad un ritardo nella consegna del prodotto o pone dei vincoli sul progetto. Per questo la metodologia Agile non elimina completamente la documentazione ma la semplifica, prediligendo un software funzionante, in quanto può essere utilizzato come un valido sistema di valutazione delle aspettative dei clienti.

3. **Preferiamo la collaborazione col cliente più che la negoziazione dei contratti:** negoziazione significa definire in anticipo i dettagli del progetto e, come visto per la documentazione, questo può portare alla nascita di vincoli. Con lo sviluppo Agile, al posto di impiegare tempo a negoziare e al dispiego di risorse, si può perseguire la via della continua collaborazione con il cliente.

4. **Preferiamo rispondere al cambiamento più che seguire un piano:** in uno sviluppo tradizionale del software, i cambiamenti, anche minimi, sono molto costosi e, perciò, si preferisce evitarli. Contrariamente, tramite il metodo Agile, le modifiche apportate al prodotto sono viste come dei feedback positivi che portano al miglioramento del prodotto e aggiungono valore al risultato finale.

Quindi, questa nuova metodica, in contrapposizione ai modelli di sviluppo tradizionali (nello specifico, al modello a "cascata"), prevede un approccio iterativo alla gestione dei progetti e allo sviluppo software, portando il team a distribuire il proprio lavoro in modo incrementale, grazie alla collaborazione tra i membri, la pianificazione continua e l'apprendimento costante.
Secondo l'approccio Agile il prodotto non viene pianificato o progettato anticipatamente, ma si gestisce il processo di sviluppo suddividendolo in iterazioni periodiche. Queste fasi, dette "sprint", della durata di due o tre settimane, permettono ad ogni componente del team di completare dei piccoli compiti scelti volontariamente, detti "task". Al termine di ogni sprint vengono analizzati i risultati ottenuti dal gruppo, in modo tale da poter verificare lo stato dei lavori e la velocità con cui il team riesce a completare le varie mansioni assegnate.

Tramite questa metodologia, al cliente è permesso ricevere dei frequenti aggiornamenti sull'avanzamento del progetto, sottoforma di rilasci di software sempre più complesso, e di poter segnalare priorità o di richiedere dei cambiamenti.

Il ciclo di vita di un prodotto sviluppato in modalità Agile è un progetto composto da 5 fasi: *Envision*, *Speculate*, *Explore*, *Adapt* e *Close*.

- **Envision:** durante questa fase vengono concordati con il cliente gli obiettivi del progetto. Inoltre, vengono scelti i componenti che andranno a formare il team di sviluppo e le norme da utilizzare per tutto il periodo lavorativo.
- **Speculate:** questo è uno stato per lo più di pianificazione; infatti si inizia ad implementare o a rivedere il progetto con le varie features, e relative stime, e i rischi che si potrebbero affrontare nel corso del lavoro.
- **Explore:** nel corso di questo ciclo viene effettivamente sviluppato il prodotto. Sono presenti attività di daily stand-up meeting (incontri di breve durata per aggiornamenti sul progresso del progetto) e di revisione frequente delle features portate a termine.
- **Adapt:** il principio alla base della metodologia Agile è la presenza di numerosi feedback che facilitano la risoluzione di problemi, se presenti. La fase di adapt è caratterizzata dalla revisione finale del prodotto da parte del cliente che lo ha richiesto e, successivamente, da una riunione con tutto il team per analizzare il lavoro concluso.
- **Close:** si chiude definitivamente il progetto e si organizza un breve incontro dove si discute delle nuove tecniche apprese per portare a termine il prodotto.

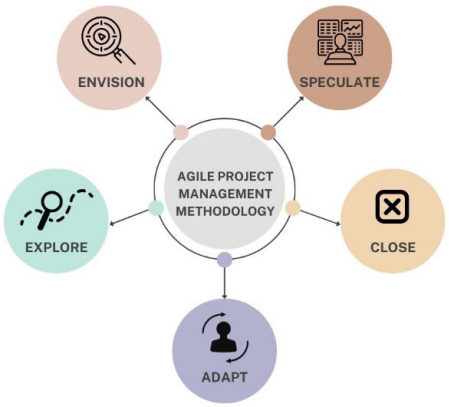

Figura 1.1: Fasi dello sviluppo Agile [43]

1.2 Team building

Le prime riflessioni riguardanti il team building hanno origine dagli esperimenti e test effettuati dallo studioso Elton Mayo sui dipendenti della Western Electric Company di Chicago negli anni '20 per analizzare come l'ambiente lavorativo influenzasse la produttività delle persone [28]. Dalla ricerca emerse che non è solo il contesto ad avere effetto sui lavoratori ma, in particolare, l'interesse delle imprese nei loro confronti e lo spirito collaborativo tra le persone di uno stesso team. Questo ha portato a sviluppare il concetto che, per ottenere risultati ottimi, bisogna lavorare sul benessere del gruppo: da qui nasce il team building.

Con il termine "*team building*" si fa riferimento al processo che sviluppa la cooperazione e la collaborazione all'interno di un gruppo. Per costruire un team efficace ed efficiente, i membri di esso devono condividere uno scopo comune, avere rispetto reciproco ed essere sempre più motivati ad utilizzare i punti di forza di ognuno per arrivare ad una soluzione ottimale. La filosofia alla base del processo di team building evidenzia come ogni membro del team ricopra un ruolo fondamentale per raggiungere gli obiettivi preposti.

Nel 1965 Bruce Wayne Tuckman nell'articolo *"Developmental Sequence in Small Groups"* [40] definisce il modello comportamentale dell'individuo sul lavoro, basato su cinque fasi: *Forming*, *Storming*, *Norming*, *Performing* e *Adjourning*. Secondo lo studioso, queste fasi sono necessarie ed inevitabili affinché un team di sviluppo affronti le sfide e i problemi di collaborazione, trovi delle soluzioni ottime, progetti in modo dettagliato il lavoro e raggiunga i risultati sperati.

- **Forming:** i membri di un gruppo sviluppano un senso di appartenenza grazie a fiducia, conoscenza e condivisione di valori comuni e alla creazione di un'immagine che rappresenti la squadra. La parte fondamentale è la scelta dei collaboratori, che viene eseguita tramite criteri tecnici (si scelgono persone con determinate competenze), funzionali (si dà importanza alle competenze trasversali e di problem solving) e relazionali (ovvero la capacità di lavorare in gruppo e di comunicare con i membri di esso). Questa selezione determina i ruoli, gli obiettivi e i compiti.

- **Storming:** in questa fase è possibile che nascano dei contrasti poichè ogni soggetto tende ad imporre la propria individualità. Il compito del leader è quello di porre fine alla crisi, mettendo in risalto gli scopi comuni da raggiungere. Il team può risolvere i conflitti in due modi: cercando di trovare un equilibrio o adottando nuove strategie lavorative.

- **Norming:** se i contrasti all'interno del gruppo vengono superati in modo efficace e positivo, si crea un senso di unità e i singoli individui riescono a sentirsi parte

di un team. Dopo di che, si iniziano a stabilire le regole, a riconoscere i ruoli, ad analizzare i metodi e gli strumenti e a creare valori comuni e obiettivi aziendali.

- **Performing:** questa è considerata come un periodo di pratica in cui il gruppo lavora in modo coeso per portare a termine gli scopi fissati. Il leader, ora, modifica il suo ruolo per far sì che la squadra sperimenti il lavoro autonomo. Un alto livello di auto-efficienza porta ad avere coraggio e fiducia nell'azienda e consente di vedere i fallimenti come un momento di crescita.
- **Adjouring:** rappresenta lo step finale, dove i compiti sono stati portati correttamente a termine e il gruppo si scioglie, creando nei singoli individui un sentimento di incertezza sul futuro, tanto da ridurre la loro motivazione. Questo momento è un'occasione per elaborare e lavorare su nuovi progetti, dando inizio ad una nuova fase di forming.

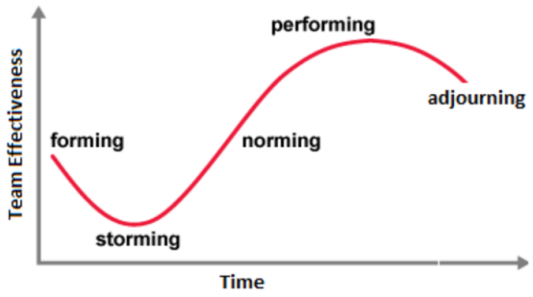

Figura 1.2: Fasi di sviluppo di un team secondo Tuckman [22]

All'interno delle imprese moderne il team building e il rafforzamento del teamwork sono considerate una priorità, in quanto la buona cooperazione degli stakeholder aziendali è indispensabile per poter migliorare le performance e la produttività. Attraverso delle attività denominate "team building exercises", i componenti di un team possono esercitarsi affinché migliori la loro creatività e la fiducia nei colleghi, ma anche a sviluppare capacità interpersonali e di leadership.

1.2.1 Cooperative Thinking

Il Cooperative Thinking (CooT) [9] è la capacità di descrivere, riconoscere, scomporre e risolvere computazionalmente i problemi in team. Quindi, l'abilità di comprendere le difficoltà e applicare, valutare e produrre una soluzione per esse sottoforma di algoritmo. Inoltre, è considerata una metodologia educativa e un approccio alla risoluzione dei

problemi che enfatizza la collaborazione, la condivisione delle idee e il lavoro di squadra. L'idea alla base è che le menti collettive possono generare soluzioni migliori e più creative rispetto alle menti individuali.

Le strutture di comunicazione rappresentano uno strumento essenziale per gli sviluppatori software in quanto aiutano a comprendere il modo in cui il software viene prodotto. Per questo motivo, quindi, è necessario educare gli sviluppatori a gestire correttamente l'organizzazione sociale del lavoro [11].

Secondo [9], si può scomporre il Cooperative Thinking in cinque principali fattori:

- **Negoziazione Complessa:** la capacità di negoziare con successo sia con gli altri componenti del team che con il cliente che richiede il prodotto, trovando delle soluzioni accettate da entrambe le parti.
- **Apprendimento Continuo:** la volontà costante di conoscere e apprendere nuovi strumenti e metodi, in modo tale da adattarsi ai cambiamenti che possono avvenire all'interno dell'ambiente lavorativo.
- **Consapevolezza di Gruppo:** l'abilità di comprendere le dinamiche che avvengono all'interno del team di sviluppo, di riconoscere i punti di forza dei singoli membri e poterli utilizzare per cooperare in modo efficace ed efficiente.
- **Organizzazione di Gruppo:** si gestisce, organizza e pianifica il lavoro da eseguire in modo ottimale, suddividendo i compiti in modo equo tra i componenti del gruppo, tenendo conto degli obiettivi da raggiungere e delle scadenze imposte.
- **Sensibilità Sociale:** vengono comprese e osservate le diversità personali e culturali dei membri del team, in modo da poter lavorare in modo rispettoso e collaborativo. Per far ciò è necessaria una comunicazione chiara e diretta, così da riuscire a creare un ambiente inclusivo.

Vari studi eseguiti su questo tema hanno evidenziato come il *Computational Thinking* e i *valori Agili* influenzano positivamente il Cooperative Thinking. Per sviluppare il CooT è necessario sviluppare le capacità acquisite grazie al Computational Thinking (CT), in modo tale da poter interagire in modo costruttivo all'interno del gruppo. L'idea alla base di questa ipotesi è che bisogna allenare i singoli ad avere un pensiero computazionale che potenzi le capacità di problem-solving.
Quindi, se il Computational Thinking è la specifica abilità di saper risolvere i problemi, i valori Agili educano gli individui a lavorare in gruppo. Questi principi offrono una vastità di punti di vista da cui si può trarre una soluzione ottimale per portare a termine compiti difficoltosi.

Il Cooperational Thinking non è però l'unione di CT e valori Agili bensì, uno strumento volto a sviluppare capacità essenziali. Recenti ricerche hanno dimostrato come l'abilità nel risolvere problemi complessi sarà una tra le competenze più ricercate nell'ambito lavorativo. Da ciò si può notare come il Cooperative Thinking abbia un enorme impatto sulla capacità di risolvere problemi complessi.

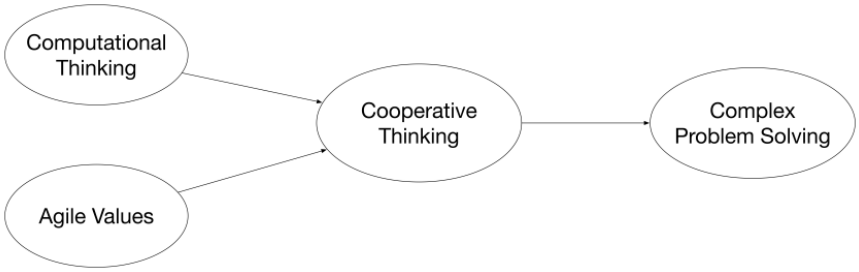

Figura 1.3: Il pensiero computazionale ed i valori agili di collaborazione sono costrutti componenti del Cooperative Thinking [9]

1.3 Team Coaching

Un team coeso e attivo porta ad avere dei vantaggi maggiori rispetto al lavoro individuale dei singoli membri. Per questo motivo, un percorso di Team Coaching è visto come uno strumento fondamentale per tutti coloro che hanno come obiettivo il rafforzamento del proprio gruppo e il raggiungimento di performance prima ritenute impensabili [17].
Quindi, il Team Coaching, come dice la parola stessa, rappresenta un allenamento, un percorso in cui si cerca di investire di più sulle performance finali e sui processi, tenendo particolarmente d'occhio il clima e la comunicazione all'interno del gruppo. Infatti, affinchè un team lavori bene, quest'ultima deve essere fluida, chiara e diretta.

Da qui nasce la figura del Team Coach, ovvero un professionista che ha il compito di aiutare un gruppo di persone a lavorare in team, insieme, massimizzando e aumentando la proficuità del risultato. Inoltre, deve supervisionare le modalità di lavoro, osservarne le dinamiche e i processi, e proporre sperimentazioni e nuove esperienze, finalizzate a promuovere la partecipazione e il coinvolgimento di tutti. Lo scopo principale è quello di non disperdere energie importanti, dando efficacia alle azioni, in modo tale da poter raggiungere gli obiettivi prefissati.
Il problema alla base di un team è individuato nella composizione stessa di esso: la squadra è formata da individui, aventi identità, personalità e metodologie diverse; se non si riesce ad integrare correttamente queste necessità tra loro, il compito da portare a termine potrebbe risentirne.

Il Team Coach deve, quindi, cercare di far combaciare tutte le diversità di una squadra, in maniera tale che esse lavorino sinergicamente verso il raggiungimento di un unico scopo comune. Uno tra i compiti principali di questa figura, infatti, è proprio quello di facilitare l'apprendimento per il singolo individuo e per il team stesso, portando ad ottenere una dinamica di squadra in cui i singoli possono mettere gli obiettivi individuali al servizio di un obiettivo di squadra, restando fedeli alle decisioni del gruppo anche se hanno delle preferenze personali differenti.

Diversi studi hanno dimostrato come la gente lavori meglio quando si diverte. A tal proposito, il Team Coaching agevola l'opportunità per i singoli e per i team di entrare in uno stato o in un flusso di lavoro efficace, riuscendo a sviluppare un sempre più crescente senso di ritmo nel lavoro che può portare ad ottenere un processo altamente efficiente e risultati eccezionali. Lavorare, e allo stesso tempo divertirsi, in team crea anche un senso di energia che ispira gli altri e consente alle persone di dare il massimo nel loro lavoro.

1.4 Serious Games

I Serious Games sono giochi che non hanno come scopo principale l'intrattenimento, ma vengono progettati per perseguire fini educativi. Infatti, sono volti a migliorare l'impegno e il lavoro dei giocatori una volta portato a termine un compito preciso, favorendo l'integrazione con meccanismi e parti del gioco, facendo diventare il compito da eseguire più coinvolgente. In più, rendono partecipe il giocatore, portandolo ad attivare una serie di processi cognitivi utili all'apprendimento e all'aumento della concentrazione [1].
Il fenomeno denominato *Gamification* fa riferimento all'utilizzo di strategie e dinamiche comuni per i giochi ma anche per attività non ludiche. Si è dimostrato come questo principio sia utile a migliorare la motivazione, ad allenare le tecniche di problem solving e ad aumentare l'abilità di valutazione di determinati eventi. Un altro termine molto diffuso oggigiorno è il *Game Based Learning*, ovvero l'apprendimento attraverso giochi o videogiochi. Questi ultimi, frutto della digital transformation, vengono considerati un mezzo d'elezione per raggiungere obiettivi formativi. Vi è una notevole differenza, però, tra Gamification, Game Based Learning e Serious Games. La prima è una metodologia in cui elementi mutuati dal gioco fanno parte di un percorso formativo, agevolando la trasmissione delle conoscenze. Il Game-based learning consiste nell'utilizzare giochi esistenti (o parte di essi) per il raggiungimento di un determinato obiettivo formativo. Contrariamente, i Serious Games non sono delle metodologie ma un prodotto attraverso cui il Game-based Learning è possibile.[25].
Vari studi hanno mostrato come l'uso dei Serious Games nell'ambito dell'Ingegneria del Software abbiano aumentato la coordinazione e la cooperazione all'interno dei team. Inoltre, l'impiego di questi giochi educativi agevola i futuri professionisti a sperimentare delle situazioni che potrebbero affrontare nello sviluppo di un software.

Figura 1.4: Serious Games [39]

Yata

Tra i Serious Games esaminati vi è Yata [31], il cui obiettivo principale è dimostrare i principi più importanti alla base di DevOps.
Per eseguire questo gioco è necessario l'ausilio di due grandi tavoli, uno destinato ai Dev e uno agli Ops. Sul piano di lavoro di quest'ultimi verrano posti gli ambienti di pre-produzione e di produzione. Lo scopo fondamentale è quello di ottenere un elevato numero di punti, fornendo maggiori funzionalità alla produzione. Questo avverrà tramite l'utilizzo di mattoncini Lego.

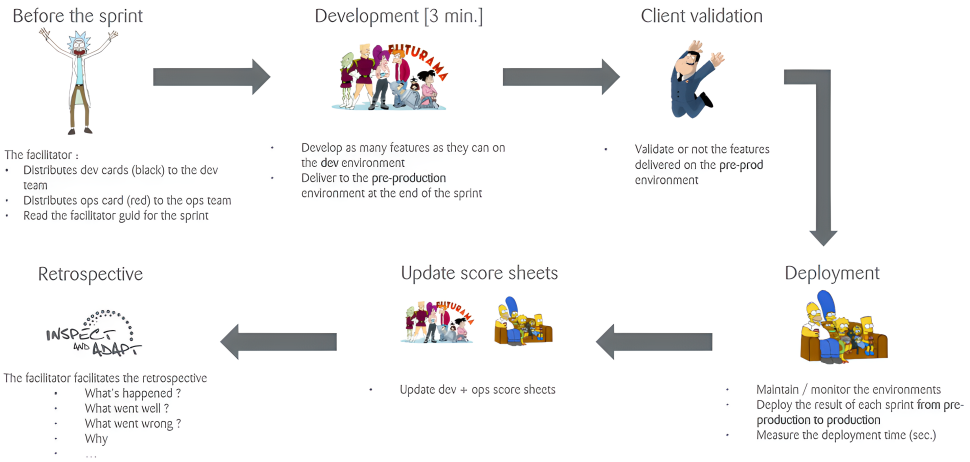

Figura 1.5: Sprint in Yata [31]

Il gioco è formato da quattro sprint da portare a termine. Durante ognuno di essi, il facilitatore deve distribuire le carte dello sprint corrispondente.

- **Sprint 1:** durante il primo sprint, i Dev hanno 3 minuti per costruire e consegnare la pre-produzione mentre gli Ops rifiutano tutte le torri non documentate. Questo viene fatto per analizzare le differenze tra l'approccio documentale e la collaborazione.

- **Sprint 2:** durante questa fase è vietato comunicare con gli Ops. I Dev devono documentare la consegna e, in più, non saranno presenti carte per loro. Lo scopo di questo sprint è quello di valutare e confrontare gli obiettivi dei Dev con quelli degli Ops (priorità, progetti, visioni).

- **Sprint 3:** successivamente, il compito dei Dev sarà quello di costruire le torri, partendo da quelle innalzate in precedenza. Inoltre, saranno messe a loro disposizione le carte dello sprint corrente. Il compito degli Ops è quello di facilitare la distribuzione di pre-produzione e produzione. Tramite ciò si vuole aumentare la collaborazione tra i membri del team.

- **Sprint 4:** in questa sessione finale i Dev devono fare lo stesso lavoro svolto nello scorso sprint, grazie anche all'aiuto delle carte della sessione attuale. Gli Ops, invece, hanno il compito di automatizzare la distribuzione tra pre-produzione e produzione. Questo sprint è volto a capire come l'automatizzazione possa facilitare e velocizzare il lavoro.

Game pipeline

Una pipeline di Continuos Delivery (CD) raggruppa tutte le attività necessarie per trasformare una modifica al codice, fatta da uno sviluppatore, in un software aggiornato che dia valore agli utenti. Tra i passaggi fondamentali troviamo la creazione di una nuova versione del software, il conseguente test di quest'ultimo ed infine la sua distribuzione. Questi passaggi possono variare in base a diversi fattori, poiché non esiste una "pipeline perfetta" che si possa adattare ad ogni contesto.
Il gioco Game Pipeline [3] ha come obiettivo principale quello di creare una pipeline utile ad un determinato scenario, andando ad ottimizzare i tempi di distribuzione. I giocatori dovranno lavorare in gruppo, discutendo quali passaggi effettuare e in quale ordine eseguirli, facendo della cooperazione e della comunicazione le chiavi principali per arrivare ad una soluzione ottima.

- **Carte scenario:** rappresentano i vari contesti possibili (es. una startup che deve creare un sito web per una banca online). Le carte scenario sono utili in quanto rendono la discussione all'interno del team più concreta e produttiva.

- **Carte pipeline step:** descrivono la struttura della pipeline. Tramite queste carte si possono scegliere quali passaggi hanno la priorità, quali possono essere facoltativi o quali si possono omettere del tutto. Inoltre, più passaggi possono essere messi in parallelo, andando a ridurre il tempo di consegna complessivo. All'interno del mazzo si possono anche trovare delle carte bianche, che rappresentano dei passaggi nuovi creati dal team.

Figura 1.6: Carte Pipeline [3]

- **Carte di vincolo:** nonostante ci sia molta libertà su come costruire la pipeline, ci sono delle regole su come ordinare i passaggi (es. non si può eseguire il deployment prima di aver creato una realease candidate). Grazie alle carte "Regole del gioco" si può tener traccia di questi vincoli. In più, bisogna stimare il tempo di esecuzione previsto, cioè quanto tempo ci vorrà per portare a termine il lavoro. Se quest'ultimo è manuale, bisogna anche tener conto del tempo trascorso ad aspettare che la persona che lo deve eseguire sia disponibile.

Lego City

Come ultimo gioco preso in esame vi è Scrum Lego City che, tramite l'utilizzo dei mattoncini Lego, si propone di simulare ogni aspetto che caratterizza la metodologia Scrum [32]. Prima di iniziare la partita, bisogna dedicare del tempo a delle attività da svolgere:

- **Fase 1:** durata stimata di 5 minuti. Si prepara la visione del prodotto da presentare al Product Owner; essa comprende l'ordinamento delle storie, l'aggiunta di nuove o il miglioramento di quelle già presenti.

- **Fase 2:** durata stimata di 5 minuti. Il Product Owner presenta la visione del prodotto e il backlog; il team può porre delle domande riguardanti le storie o i requisiti.

In seguito, si può avere una breve riunione (all'incirca di 5 minuti) per pianificare il rilascio del prodotto, dove il team si concentra sulla stima di più storie possibili, in modo tale da ottenere un'idea iniziale dei costi di rilascio.
Fatto ciò, si inizia facendo il primo incontro di pianificazione dello sprint, in modo tale da essere preparati ad affrontare i 4 sprint presenti per portare a termine il maggior numero di storie contenute nel backlog. Alla fine di ogni sprint, viene fatta una riunione dove il team mostra al Product Owner cosa è stato costruito. Al termine, si esegue la retrospettiva con tutto il gruppo.
La partita avrà una durata di circa 40 minuti. Alla fine del gioco, il Product Owner effettua la Release Review ed analizza con il team i risultati ottenuti.

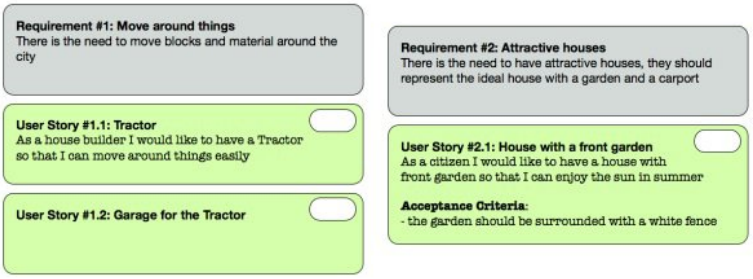

Figura 1.7: Carte Lego City [32]

Capitolo 2

Scrum

In questo capitolo, si porrà attenzione al mondo di Scrum, un framework ampiamente adottato nel mondo Agile e della gestione dei progetti. Si esploreranno i ruoli chiave all'interno di Scrum, ossia il Product Owner, lo Scrum Master e i Developers, così come i fondamentali artefatti che guidano il processo.
Il Product Owner sarà al centro dello studio poiché è il responsabile della definizione delle priorità, della gestione del backlog e della garanzia che il prodotto soddisfi completamente le esigenze dei clienti. Inoltre, si darà uno sguardo approfondito al ruolo ricoperto dallo Scrum Master, il facilitatore del processo Scrum. Scopriremo come il suo impegno nel rimuovere gli ostacoli e nel garantire una rigorosa adesione ai principi di Scrum favorisca la trasparenza e la produttività all'interno del team. I Developers, ovvero il terzo pilastro del framework, saranno oggetto di analisi per comprendere come il loro lavoro collaborativo e la loro responsabilità condivisa contribuiscano alla consegna di prodotti di alta qualità.
Infine, si analizzeranno gli artefatti di Scrum, come il Product Backlog, lo Sprint Backlog e l'Increment, e scopriremo come questi strumenti essenziali rappresentino una struttura chiara per la pianificazione, la consegna e la revisione dei risultati.

2.1 Framework Scrum

Scrum è un framework definito nel 1995 da Ken Schwaber and Jeff Sutherland [5], utile alla gestione dei progetti Agile. Permette ad un "piccolo" team di strutturare e controllare il proprio lavoro attraverso un insieme di valori, principi e pratiche. Questo framework, quindi, spinge il team ad imparare attraverso l'esperienza maturata durante il conseguimento di un lavoro, ad organizzarsi in modo autonomo e a riflettere sui risultati conseguiti e sugli insuccessi per poter migliorare continuamente.
Spesso si tende a confondere Scrum con Agile, in quanto Scrum ha alla base il miglioramento costante, che è uno dei principi fondamentali di Agile. In realtà, il framework

consente di effettuare e controllare l'avanzamento del lavoro, mentre la metodologia Agile permette il miglioramento continuo tramite piccoli e frequenti rilasci del prodotto.
Scrum è basato sull'empirismo e sul pensiero Lean [38]. Secondo l'empirismo, la conoscenza deriva dell'esperienza e le decisioni devono essere prese in funzione di quello che si osserva. Il pensiero Lean, invece, suggerisce di ridurre gli sprechi per focalizzarsi su ciò che risulta essere essenziale.
Inoltre, Scrum è un framework euristico, ovvero si basa sull'apprendimento continuo e sul sapersi adattare a situazioni variabili e mai uguali tra loro. Si consideri che all'inizio di un progetto il team non dispone di tutte le conoscenze necessarie per portarlo a termine, ma queste verranno acquisite pian piano con l'avanzare dei lavori. La struttura di Scrum, infatti, permette ai team di adeguarsi di continuo ai cambiamenti e alle esigenze dei clienti, tramite una riconsiderazione delle priorità assegnate e dei cicli di rilascio.
Il framework Scrum presenta una struttura ma non risulta troppo rigido, infatti la sua esecuzione si adatta senza problemi a qualsiasi tipo di organizzazione.

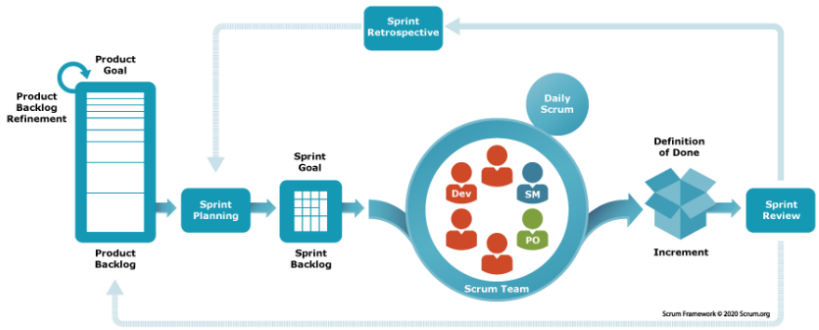

Figura 2.1: Framework Scrum [33]

Un team Scrum è in generale composto da poche persone (di solito da 3 a 7, e comunque meno di 10), sufficienti per completare una grande mole di lavoro in un breve periodo, detto sprint. All'interno del team è necessario che vi siano tre ruoli fondamentali: Product Owner, Scrum Master e il team di sviluppo, che include, oltre agli sviluppatori, tester, progettisti e tecnici.

2.1.1 Product Owner

Il Product Owner (PO) è la figura che si interfaccia con il cliente ed, eventualmente, con gli stakeholders rappresentandoli nel team (raccoglie, infatti, le loro esigenze per poi trasmetterle al team di sviluppo). Gestisce il flusso di lavoro in ingresso scrivendo, selezionando, priorizzando e raffinando le User Stories contenute nel Product Backlog.
L'obiettivo del suo lavoro è quello di portare a termine il prodotto avente valore massimo entro le scadenze stabilite.

Questo vuol dire che il risultato sviluppato deve fornire valore agli stakeholder risolvendo tutte le esigenze rappresentate nelle User Stories.

Figura 2.2: Attività principali del Product Owner [42]

Il Product Owner non può, però, essere l'unico responsabile per tutto il lavoro. L'intero team deve essere consapevole di dover essere il più produttivo possibile e di dover migliorare le proprie pratiche. Quindi, il team deve decidere la mole di lavoro da portare a termine in uno sprint e cercare di produrre un incremento di prodotto di valore massimo. Tuttavia, il Product Owner ricopre una posizione unica: è infatti la persona più vicina al lato business del progetto, colui che viene incaricato dall'azienda di rilasciare il prodotto e da cui ci si aspetta che faccia tutto il necessario per soddisfare le esigenze degli stakeholders.

La mansione principale del PO è quella di gestire correttamente il Product Backlog, assicurandosi che risultino visibili sia quest'ultimo che gli avanzamenti realizzati durante la sua lavorazione. Inoltre, è il responsabile di cosa il team dovrebbe fare e di cosa, invece, si può evitare di svolgere, facendo una scelta tra l'estensione delle funzionalità e le tempistiche di realizzazione, per arrivare ad ottenere il migliore prodotto possibile. La gestione del Product Backlog non è una scienza ma, anzi, può essere vista come un'arte che dipende fortemente dall'individuo e dall'ambiente lavorativo in cui questo è posto. Perciò, non esiste un framework che descriva in maniera rigorosa una struttura ottima per il corretto svolgimento del lavoro del Product Owner.

Solitamente, si tende a valutare un progetto tramite il *Triangolo di Ferro* (o anche *Triangolo dei Vincoli*) che permette di misurare il successo di un lavoro attraverso tre parametri: costo, tempo e scopo - si veda la Fig.2.3. Tali parametri, però, vengono spesso visti come dei vincoli più che delle variabili da configurare.

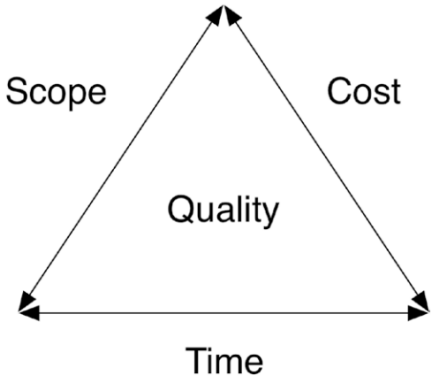

Figura 2.3: Triangolo di Ferro [24]

Per un Product Owner che deve gestire al meglio lo sviluppo di un prodotto, il lato "scopo" rappresenta la qualità e la completezza del lavoro da portare a termine. Il PO è la figura responsabile della definizione e della prioritizzazione dei requisiti, in modo da portare il team a concentrarsi sulle funzionalità più cruciali e di maggior valore per il cliente. Questo necessita di una comprensione approfondita e dettagliata delle esigenze degli utenti e l'abilità di arrivare a prendere decisioni chiare e informate sulle features da sviluppare. Il "tempo" indica la durata dei diversi sprint e il ritmo complessivo del processo di sviluppo. L'operato del Product Owner è volto a garantire che le esigenze del cliente vengano rispettate e consegnate rapidamente, rispettando, però, gli intervalli di tempo previsti per ogni sprint. Infine, la parte relativa al "costo" indica le capacità e le risorse disponibili per portare a termine il prodotto. Per far ciò, è necessaria la collaborazione diretta tra il Product Owner e il team di sviluppo, che lavorano a stretto contatto per comprendere le risorse disponibili e per gestire le aspettative degli stakeholder in merito a ciò che si può costruire con i mezzi a disposizione.

La sfida del Product Owner è quindi quella di riuscire ad ottenere una soluzione che rispetti questi limiti, rientrando sempre nel budget stanziato. Lo scopo, infatti, non è quello di costruire il prodotto ottimale, ma il prodotto che meglio soddisfi i requisiti posti [24].

2.1.2 Scrum Master

Lo Scrum Master può essere paragonato ad un "servant leader" che aiuta il resto del team di sviluppo a portare a termine il processo piuttosto che assumere una posizione autoritaria. Per questo motivo deve, quindi, avere una buona conoscenza del framework Scrum, in modo da poter istruire gli altri nelle sue sfumature.
Questa figura ha il compito di promuovere un ambiente di lavoro cooperativo e aperto,

incoraggiando la comunicazione trasparente all'interno del gruppo. In più, è anche responsabile della rimozione di eventuali ostacoli che possono contrastare il progresso del team, garantendo che il flusso di lavoro sia continuo e senza impedimenti.
Infine, collabora con il Product Owner per la creazione del Product Backlog, assegnando valore e priorità alle varie task. Inoltre, aiuta il gruppo ad individuare ed adottare le tecniche utili allo sviluppo di un prodotto compiuto e funzionante al termine di ogni sprint.

Lo Scrum Master si occupa, quindi, di organizzare riunioni stand-up e di pianificazione dello sprint, così da evitare al team compiti eccessivamente difficili o non essenziali, di revisionare lo sprint e di redigere le retrospettive, in modo tale da riuscire a tenere nota di ciò che può essere migliorato e degli elementi utili alla realizzazione degli sprint successivi. Inoltre, predispone incontri individuali con i singoli membri del team, affinchè si possano trovare eventuali punti di disaccordo nel gruppo, e di stilare il report, eseguendo un'analisi periodica dei grafici di burn-down per capire cosa viene prodotto e con quale cadenza. Quindi, questa figura gioca un ruolo di fondamentale importanza nel supporto del team e nel garantire che il processo Scrum sia implementato in modo efficace, portando il team a migliorare le prestazioni nel corso del tempo.

Figura 2.4: Attività dello Scrum Master [42]

2.1.3 Team di sviluppo

In un team Agile, il team di sviluppo (o Developer) sono i professionisti la cui mansione è quella di sviluppare e rilascire un incremento del prodotto. Per raggiungere questo scopo, il gruppo si auto organizza, scomponendo il lavoro in sotto compiti, ognuno assegnato ad un membro della squadra.

Il framework Scrum prevede che il team sia cross-funzionale, ovvero composto da individui che possiedono le diverse competenze necessarie per produrre un prodotto di valore massimo. I developer partecipano attivamente agli eventi che caratterizzano Scrum, co-

me, ad esempio, le riunioni di pianificazione degli sprint, in cui discutono e stimano il lavoro da portare a termine durante lo sprint successivo.
All'inizio di ogni sprint, il gruppo riceve i compiti e le indicazioni dal Product Owner; sta poi ai membri del team autogestirsi per decidere chi deve portare a termine una determinata task. Per questo motivo, i team devono essere multifunzionali, cioè ogni componente ha delle competenze trasversali su vari ambiti del lavoro.
Al termine dello sprint, il team mostra i lavori ultimati e suggerisce modifiche e miglioramenti da apportare per incrementare il valore del prodotto.

Oltre a svolgere un ruolo tecnico, il team di sviluppo è coinvolto nelle attività di revisione del codice. Gli sviluppatori collaborano tra loro per migliorare la qualità del codice e condividendo conoscenze. La collaborazione rappresenta un aspetto fondamentale del lavoro dei developer: quest'ultimi lavorano a stretto contatto con il Product Owner, in modo che sia garantita una comprensione chiara dei requisiti del prodotto, e con lo Scrum Master, affinché vengano affrontati eventuali ostacoli o impedimenti che potrebbero rallentare lo sviluppo del compito.

Figura 2.5: Attività del Team di sviluppo [42]

2.1.4 Artefatti Scrum

Gli artefatti Scrum sono le varie informazioni che un team usa per descrivere dettagliatamente il prodotto che si sta sviluppando, gli strumenti che occorrono per portarlo a termine e i compiti eseguiti durante il progetto [41]. Questi vengono creati durante le attività alla base di uno sprint e tra i principali artefatti vi sono:

- **Backlog di prodotto:** è un elenco di nuove funzionalità, miglioramenti, task o requisiti fondamentali da eseguire. Viene compilato seguendo la domanda del cliente, l'analisi della concorrenza e la relativa richiesta all'interno del mercato. Il backlog del prodotto è in continuo aggiornamento, in quanto viene stilato ogni volta che si hanno a disposizione delle nuove informazioni. Infatti, viene gestito dal Product Owner tra uno sprint e il successivo, in modo tale da poter inserire task

che si trovavano in sprint attivi, ma la cui priorità si è ridotta, spostandoli quindi nel backlog del prodotto.

- **Backlog dello sprint:** è costituito dall'insieme di task facenti parte del backlog del prodotto che sono state scelte per essere sviluppate nello sprint successivo. Il backlog dello sprint viene creato dal team per pianificare il rilascio degli incrementi futuri e per descrivere il lavoro necessario a creare l'incremento. Per fare ciò, vengono suddivise le task selezionate in lavori più piccoli e facilmente realizzabili.

- **Incrementi:** è formato dai rilasci al cliente, ottenuti portando a termine le task del backlog di prodotto durante uno sprint. Per ogni sprint c'è sempre un incremento, deciso durante la fase di pianificazione. Avere degli incrementi rappresenta un vantaggio per il team di sviluppo in quanto si riesce a mantenere tutto il lavoro allineato agli elementi che costituiscono il backlog.

Oltre agli artefatti Scrum sopra esposti, esistono alcuni artefatti estesi o meta artefatti. Sebbene non siano ufficiali poichè non presenti all'interno delle linee guida di Scrum, questi artefatti estesi aggiungono valore e ulteriori informazioni ai cicli di sviluppo.
Tra questi meta artefatti troviamo:

- **Grafico burn down:** viene spesso usato per monitorare l'avanzamento verso lo scopo pianificato per lo sprint. I grafici burn down (o burn up) mostrano le task completate e la velocità di esecuzione del team, in modo tale da verificare se quest'ultimo sia in grado di ultimare tutto il lavoro organizzato o se è necessario riassegnare le priorità ai vari compiti [2].

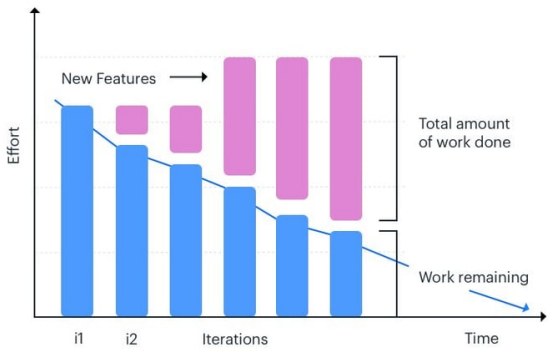

Figura 2.6: Grafico Burn Down [30]

- **Definition of Done:** illustra le attività da portare a termine affinché i requisiti (ovvero i product backlog item) che caratterizzano uno Sprint siano visti come "completati", in linea con i criteri di qualità richiesti per il prodotto, e ultimati in modo tale da essere considerati come parte di un incremento [36]. Seguendo quanto riportato nella guida di Scrum [34], che fa riferimento alla Definition of Done come un "commitment" del Product Increment, un incremento di prodotto si ha quando un requisito (espresso nella forma di User Story), soddisfa la Definition of Done.
  Quindi, possiamo dire che la DoD facilita la trasparenza all'interno del team e, in particolar modo, garantisce un livello minimo di qualità del prodotto. Inoltre, essa include tutto il necessario che l'azienda deve tenere in considerazione per poter rilasciare al cliente il prodotto.

2.1.5 User Story

Una pratica molto comune in Scrum (ma anche in altri framework come Extreme Programming) è scrivere i requisiti di un prodotto sotto forma di *User Story*. Queste sono una definizione ad alto livello dei requisiti e features che formeranno il prodotto. Di solito, si tende a scrivere le User Stories immaginando un discorso con il cliente, che richiede e desidera una nuova funzionalità. La quantità di informazioni scritte in una User Story è lo stretto necessario affinché il team possa effettuare una stima ottimale per calcolare l'effort richiesto per la realizzazione [10].

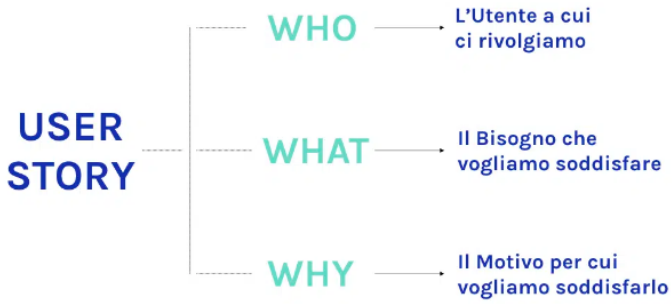

Figura 2.7: Template User Story [12]

L'acronimo INVEST è stato coniato da Bill Wake nel 2003 [27] ed è uno strumento utile ai Product Owner per tenere a mente una serie di criteri volti a valutare la qualità di una User Story. Infatti, se la story non soddisfa pienamente uno di questi vincoli, il team potrebbe decidere di riformularla o di riscriverla completamente [7].
Quindi, delle buone User Stories dovrebbero essere:

I: Indipendenti, poiché se esse non risultano essere indipendenti tra loro possono provocare stime non corrette, vincoli di implementazione e rigidità nella pianificazione

dei rilasci. Quindi, per evitare ciò, è preferibile segmentare le User Stories in modo tale da poterle implementare non seguendo un ordine prestabilito.

N: Negoziabili, in quanto le User Stories, per loro natura, sono brevi descrizioni delle funzionalità desiderate e quindi negoziabili con il cliente.

V: Di valore, idealmente tutte le User Stories sono di valore per il prodotto finale ma non è sempre così. L'importante, però, è evitare di sprecare tempo (e risorse) in storie di poco valore e provare a trovare dei benefici utili alle storie tecniche.

E: Stimabili, in quanto nella metodologia Scrum, la stima di una storia e del tempo necessario per svilupparla è un'attività fondamentale all'interno del team.

S: Della giusta dimensione, in modo tale che esse non risultino essere troppo grandi o troppo piccole. Una User Story non ha una dimensione fissa, pertanto, nel caso una storia sia troppo ampia, è possibile dividerla in sotto storie o, nel caso invece sia scarsa, si possono combinare più User Stories affinchè l'effort complessivo comporti almeno una giornata lavorativa.

T: Testabili, perché il testo di ciascuna User Story deve includere un criterio di accettazione misurabile, in modo tale che l'accettazione sia verificata tramite test. Quando i test (che si preferisce automatizzare il più possibile) passano, allora si dimostra che la storia è completamente "fatta" (*done*).

A causa della natura volatile dei requisiti e dei continui cambiamenti a cui questi sono sottoposti, spesso assegnare la priorità alle User Stories rappresenta un problema. Il processo di prioritizzazione è un meccanismo complesso e laborioso, che non favorisce il coinvolgimento di tutti i membri del gruppo. In questo contesto, la Gamification può essere adottata come strategia volta a migliorare questo processo in quanto promuove esperienze analoghe a quelle proposte dai giochi.
Seguendo quanto riportato nell'articolo "*PRIUS: Applying Gamification to User Stories Prioritization* [26]" si può fare uso del modello PRIUS per assegnare le varie priorità in un processo di sviluppo Agile. Questo approccio combina la tecnica di prioritizzazione dei requisiti con elementi chiave presenti nei giochi, al fine di favorire il lavoro degli stakeholders durante le attività.
Da uno studio effettuato si è potuto notare come, i gruppi che utilizzavano la versione integrata con la Gamification abbiano partecipato più attivamente nell'operazione di prioritizzazione delle User Stories (circa del 92%) rispetto a quelli che non l'hanno impiegata (che hanno dato priorità alle stories per circa il 75%). Da ciò si evince come l'uso del modello PRIUS abbia positivamente influenzato questo processo, portando ad avere una media dei commenti per User Stories più alta rispetto ai livelli normali.

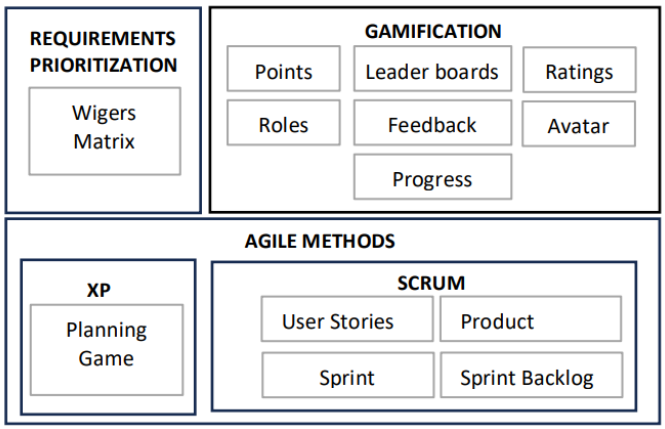

Figura 2.8: Modello PRIUS [26]

2.1.6 Sprint

Si definisce "sprint" un periodo di tempo (di solito breve: per esempio due settimane) in cui il team collabora e coopera per produrre un incremento di funzionalità del prodotto. Queste sessioni, alla base della metodologia Agile e del framework Scrum, se ben progettate, possono portare a migliorare il rilascio del software e a ridurre le problematiche interne al team.
Ogni sprint dura da 2 a 4 settimane, e dovrebbe terminare col conseguimento del Product Goal. Se la durata di uno sprint fosse pù lunga si rischierebbe che l'obiettivo da conseguire perda di validità e di aumentare i rischi, mentre, mantenendo la durata breve degli sprint si possono generare più cicli di apprendimento e verificare più facilmente i costi di produzione.

Durante questo breve periodo di tempo, il team si impegna a completare una serie di obiettivi e piccole attività definiti durante la fase di pianificazione. L'identificazione di questi avviene grazie alla collaborazione costante tra il Product Owner, ovvero la figura che rappresenta il cliente e i suoi requisiti, ed il team di sviluppo. Al termine dello sprint, i developer prendono parte ad una riunione di revisione in cui vengono mostrate al PO (eventualmente anche agli stakeholders)le funzionalità portate a termine. Inoltre, viene effettuata anche una retrospettiva dello sprint stesso, durante la quale il team riflette sulle prestazioni, identifica eventuali aree di miglioramento e pianifica le azioni future.

2.1.7 Pianificazione dello sprint

La pianificazione dello sprint dà il via allo sprint stesso. Lo scopo di questa attività è la definizione di cosa può essere attuato nel periodo indicato e come verrà realizzato tale

lavoro, ovvero di quali strumenti si avrà bisogno. Essa viene ideata durante una breve riunione dove è presente l'intero team di sviluppo.
La pianificazione dello sprint tratta tre argomenti:

- **Perché questo Sprint è di valore?** Il Product Owner ha il compito di aumentare il valore del prodotto nell'arco di uno Sprint e l'intero team deve collaborare per fissare uno Sprint Goal che possa far risaltare la qualità aggiunta in questa sessione.
- **Cosa si può fare in questo Sprint?** Scegliere cosa può essere ultimato in uno sprint può risultare complesso. Più il team lavora sulle proprie performance, più si riesce a selezionare la quantità di lavoro da poter completare in una sessione.
- **Come si svolgerà il lavoro scelto?** Per ogni task da portare a termine, il team di sviluppo suddivide i vari compiti in parti più piccole che possano durare anche meno di una giornata. Questa attività è a pura discrezione dei membri del team e nessuno può interferire con essa.

L'intera riunione, quindi, è incentrata sulla collaborazione e sull'assunzione di responsabilità condivise per il successo dello sprint. Alla fine, si avranno degli obiettivi chiari e ben delineati, le attività saranno comprese ed il team sarà pronto a portare a termine il lavoro.

2.1.8 Retrospettive

La Retrospettiva è una riunione di poche ore che rappresenta l'atto finale dello sprint. In questo incontro il team revisiona l'ultimo sprint, focalizzandosi sulle iterazioni, i processi e gli strumenti adoperati per portarlo a termine. La retrospettiva, inoltre, permette di identificare i potenziali rischi, eliminare errori commessi in precedenza, ma anche di apportare un miglioramento continuo nel gruppo. Mira, quindi, a favorire un umbiente aperto e collaborativo in cui i membri del team si sentano liberi di esprimere le loro opinioni e condividere le loro esperienze.
Alla restrospettiva partecipano i componenti del gruppo, lo Scrum Master e Product Owner. Di fondamentale importanza è la presenza di quest'ultimo in quanto fornisce una prospettiva orientata al cliente, garantendo che le decisioni prese siano in linea rispetto agli obiettivi dell'azienda e alle esigenze dell'utente.
Durante questa riunione, il team può utilizzare varie tecniche per facilitare la discussione. Un modello spesso utilizzato è lo "Start, Stop, Continue", in cui si considera cosa il gruppo dovrebbe iniziare a fare, cosa dovrebbe smettere di fare e cosa, invece, dovrebbe continuare a fare per migliorare le prestazioni [18]. Inoltre, vengono esplorati specifici aspetti del processo di sviluppo, come la collaborazione e la comunicazione, gli strumenti adoperati e la gestione dei rischi o degli impedimenti. L'obiettivo di ciò è l'identificazione di azioni concrete che si possono implementare per accrescere l'efficacia del lavoro di squadra.

Capitolo 3

Giochi per la formazione del Product Owner

Questo capitolo è dedicato all'uso dei Serious Games nell'educazione e nella formazione dei Product Owner. In questo capitolo andremo ad analizzare tre giochi particolarmente efficaci: Escape the Boom, Scrumble e PO Value Game.
Mentre il ruolo del Product Owner continua a guadagnare sempre maggior importanza nell'ambito della gestione dei progetti e dello sviluppo dei prodotti, la formazione di questi professionisti è diventata fondamentale. Questi giochi, progettati appositamente per affinare le competenze del Product Owner, offrono un approccio coinvolgente e pratico all'apprendimento.
Escape the Boom, Scrumble e PO Value Game rappresentano strumenti ludici volti ad aiutare i futuri Product Owner a sviluppare una profonda comprensione dei principi chiave di Scrum, della gestione del backlog e dell'interazione con gli stakeholder. Ci focalizzeremo sul funzionamento e lo svolgimento di essi, su quali sono i loro obiettivi e come contribuiscono a sviluppare le abilità critiche del Product Owner.

3.1 Escape the Boom

Escape the Boom è un gioco collaborativo ideato per smartphone. L'operatore deve disinnescare la bomba visualizzata sul proprio telefono, ma non sa come procedere. La squadra di supporto ha un manuale e può capire cosa fare, ma non vede la bomba. Quindi, per cercare di non far esplodere l'ordigno, è necessario che le due parti comunichino molto e chiaramente. Il ritrovamento e disinnesco della bomba deve avvenire in meno di cinque minuti.
La partita può essere effettuata dal gruppo sia in una stessa stanza, quindi di persona, che da remoto, con i giocatori collegati a distanza. È possibile scaricare l'applicazione

gratuitamente e, per giocare insieme al proprio team, è necessario averla scaricata su un solo dispositivo.

Figura 3.1: Escape the Boom [13]

3.1.1 Adattamento del gioco

Oltre ad essere un gioco da poter fare con la famiglia e gli amici, Escape the Boom è un ottimo strumento per attività di team building. Ad esempio potrebbe essere utilizzato in corsi di formazione che coinvolgono canali di comunicazione a distanza o retrospettive agili. In più, permette di riflettere sui modelli di comunicazione e collaborazione all'interno di un team quando questo è sotto pressione [14].
Per questi motivi, ho pensato di adattare Escape the Boom alla metodologia Agile e allo sviluppo Scrum, facendolo diventare una valida alternativa a Scrumble. Per fare ciò, prima di tutto, ho dovuto studiare nel dettaglio i ruoli presenti all'interno dei team Agili, in modo tale da poterli inserire nel gioco. Da qui derivano:

- **Product Owner:** è l'artificiere, ovvero l'unica persona che può guardare la bomba. Prima di iniziare la partita, deve scaricare sul proprio dispositivo l'applicazione.
- **Scrum Master:** è colui che ricopre il ruolo di intermediario tra il team e il Product Owner, in quanto non ha accesso al manuale e non può vedere la bomba, il suo compito principale è quello di riferire le informazioni ottenute dal team al Product Owner e viceversa.
- **Team:** sono le sole persone a poter leggere il manuale per poter trovare una soluzione che impedisca alla bomba di esplodere.

3.1.2 Fasi di Escape the Boom

- **Pre Game:** dopo aver scaricato l'applicazione sul proprio dispositivo, il Product Owner ha il compito di effettuare una stima quantitativa del numero dei livelli che

il team potrebbe essere in grado di superare e del tempo che impiegheranno per portare a termine la partita.

- **Livelli:** per creare degli sprint simili a quelli presenti in Scrumble, ho deciso di strutturare il gioco su quattro livelli, ognuno della durata di 5 minuti.
  Alla fine di ogni livello, Il Product Owner deve riportare per iscritto il tempo impiegato dal team per trovare una soluzione che possa disinnescare la bomba o, nel caso in cui il team non riesca, l'esplosione della stessa.

- **Debito Tecnico:** se il team non è riuscito a collaborare e cooperare al fine di trovare il modo per risolvere l'enigma, il tempo previsto per evitare l'esplosione della bomba diminuirà di un minuto, passando, quindi, da 5 minuti a 4 minuti.

- **Fine del gioco:** Il gioco termina quando viene eseguito l'ultimo livello, ovvero il quarto. A questo punto, il Product Owner farà la somma del tempo impiegato per superare tutti i livelli e la confronterà con la stima effettuata prima dell'inizio della partita: se il tempo effettivo risulta essere minore o uguale al tempo previsto allora il team ha vinto.

3.2 Scrumble

Scrumble [37] è stato ideato nel 2014 da Romain Trocherie, coach Agile e Scrum Master presso Pyxis Suisse. Il gioco è un modello dei ruoli, degli artefatti e degli eventi Scrum. Inoltre, consente di mettere in evidenza i vari problemi che possono nascere all'interno di un team e come questi vengono risolti, il tutto in un ambiente informale.
La durata di una partita è all'incirca di due ore e i giocatori ricoprono i ruoli di:

- **Product Owner:** guida e consiglia il team per incrementare il valore del prodotto;

- **Scrum Master:** aiuta a facilitare il progetto e le interazioni tra i membri del team. Non è parte attiva del gioco;

- **Team di sviluppatori:** i membri del team si auto-organizzano per creare il prodotto con valore massimo.

Per giocare a Scrumble si dispone di un tabellone, di un dado e di una pedina che rappresenta il team di sviluppo. In più, sono presenti tre mazzi di carte: le carte problemi, le carte giornaliere e le carte dedicate alla revisione.
Durante la partita, il successo dei giocatori dipende dai punti valore totalizzati, ottenuti portando a termine ogni sprint. Bisogna fare in modo di massimizzare il numero di punti valore, tramite sfide sia individuali che collettive. In Scrumble non ci sono specifiche condizioni che conducono alla vittoria. Infatti, il successo dei giocatori viene valutato considerando il numero di sprint portati a termine.

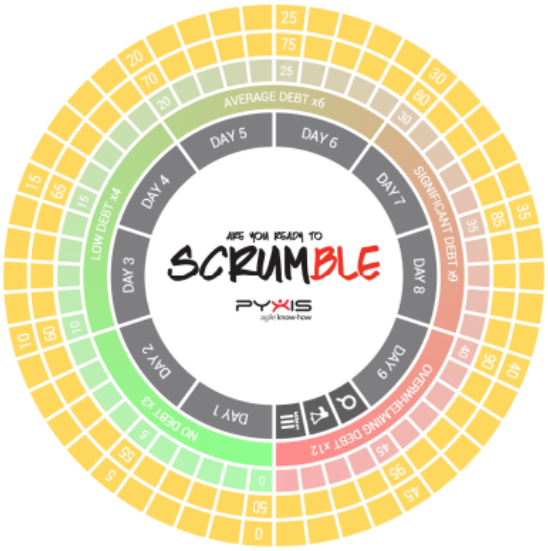

Figura 3.2: Tabellone Scrumble [37]

3.2.1 Fasi di Scrumble

Scrumble è suddiviso in fasi da portare a termine, quali: Pre-sprint, Sprint, Sprint Planning, Development Days, Sprint Retrospective, Sprint Review.

- **Pre-sprint:** Dopo aver spiegato le regole del gioco, il Product Owner presenta al team il prodotto da creare durante la partita. Il Product Owner, quindi, distribuisce al team 15 User Stories e alloca 45 punti valore ad esse: più punti vengono assegnati ad una User Story, più essa risulta di grande valore per la soddisfazione del cliente. Una volta fatto ciò, i membri del team devono stimare la relativa complessità dei compiti e scegliere quali portare a termine durante lo sprint.

- **Sprint:** Lo sprint comprende una sessione di pianificazione, nove turni di sviluppo, una sessione di revisione e una retrospettiva.

- **Sprint Planning:** Durante la pianificazione dello sprint, i membri del team scelgono il numero di attività da eseguire. Le User Stories scelte vanno a formare lo Sprint Backlog.

- **Development Days:** I membri del team (esclusi Product Owner e Scrum Master), dopo aver scelto se eseguire le attività o ridurre il debito tecnico, lanciano un dado. Se il giocatore sceglie di ridurre il debito tecnico, porta indietro la propria pedina fino al cerchio corrispondente ad esso. Se invece sceglie di eseguire le attività, muove in avanti la pedina in base al numero uscito sul dado.

Se il numero uscito sul dado è 1, si può ritirare il dado. Se il numero uscito sul dado è 6, il giocatore pesca dal mazzo una carta problema e la legge al team. Una carta problema rappresenta una complicazione che tutto il team deve affrontare.

Se tutte le task vengono portate a termine prima della fine del giorno 9, il team sceglie una delle seguenti opzioni:

1. Aggiungere una/più User Stories allo Sprint Backlog;
2. Pagare il debito tecnico;
3. Terminare lo sprint. Questo si può fare solo se non è rimasta nessuna User Stories nel product backlog.

- **Sprint Retrospective:** Quando si arriva alla casella "Retrospettiva", il Product Owner fa un'analisi di cosa è stato fatto durante lo sprint, valutando lo stato di avanzamento del team. Se rimangono User Stories non ancora portate a termine, il team ottiene un malus: vengono assegnati un numero di punti di debito tecnico pari alle task non effettuate.
- **Sprint Review:** Se ci si trova sulla casella nel secondo cerchio grigio, vengono concessi due minuti ad ogni giocatore per esprimere cosa ha imparato nel corso della partita o cosa non gli è stato utile.

Debito Tecnico

Il debito tecnico rappresenta un qualsiasi tipo di complicazione che può presentarsi all'interno di un progetto, ed è correlato a task non portate a termine, incomplete o poco adeguate al contesto di sviluppo: può essere un bug, un errore nei test effettuati, problemi di interazione o di cooperazione all'interno del team e via dicendo.

Points of debt	0 to 9	10 to 19	20 to 29	30 to 39	40 to 49
Debt factor	x 3	x 4	x 6	x 9	x 12
Handicap	- 25%	0%	+ 50%	+ 125%	+ 200%

Figura 3.3: Fattori che producono debito tecnico [37]

All'inizio del gioco, vi è un fattore prestabilito che vale x4 e l'indicatore del debito è di 15 punti. Durante la partita, però, il valore può aumentare o diminuire, a seconda delle difficoltà incontrate o delle decisioni prese dai membri del team. Quando l'indicatore del debito raggiunge i 49 punti, la partita termina.

3.3 Product Owner Value Game

Product Owner Value Game [15] è un gioco di carte utile alla formazione dei PO, in quanto mostra come bilanciare il valore consegnabile al cliente (quantificato in Business Value Point bvp) con le risorse a loro disposizione (quantificate in story point spendibili per realizzare ogni story point). In più, il gioco ha un meccanismo basato su iterazioni di raffinamento del backlog.
L'obiettivo del gioco è di imparare il valore delle scelte che fa il Product Owner quando deve sistemare le features (dette anche epiche) o le User Stories di un backlog, portando a massimizzare il valore consegnato al cliente finale prima della fine dei cicli. Il gioco, quindi, coinvolge i partecipanti nella simulazione di un contesto di sviluppo di prodotto. I gocatori assumeranno il ruolo del Product Owner e dovranno gestire un backlog di elementi, ognuno dei quali ha un valore specifico per il cliente. Gli elementi del backlog sono descritti in termini di benefici, costi e complessità.
I partecipanti hanno il compito di valutare e assegnare priorità a questi elementi in base a vari criteri, come il ritorno sull'investimento, il rischio, la complessità e altri fattori rilevanti. Il gioco mette alla prova la capacità dei giocatori di prendere decisioni informate sulle priorità e di comunicare le scelte fatte in modo chiaro e convincente.
Il numero degli sprint da effettuare non è scelto a priori, ma dal lancio di un dado da usare ad ogni sprint.
Ogni ciclo inizia con 20 story point: questi rappresentano la velocità, ovvero la quantità di lavoro che riesce a fare il team in ciascuna iterazione.

Figura 3.4: Carte del Product Owner Value Game [15]

3.3.1 Fasi di Product Owner Value Game

Lo sprint inizia con una capacità di realizzazione di 20 punti storia. Quindi, il Product Owner deve decidere se:

- Raffinare delle features in User Stories, da aggiungere poi al backlog. Questa scelta costa 2 punti storia.
- Raffinare una User Story. Per far ciò paga mezzo punto storia.
- Fornire valore prodotto al costo della User Stroy.
- Continuare fino all'esaurimento dei 20 punti storia dello sprint.

A questo punto si tira il dado, sommando il risultato del tiro al numero di sprint corrente: se il numero che esce è maggiore di 7 allora il gioco termina, altrimenti si inizia un nuovo sprint.
Alla fine di ogni iterazione viene effettuata una retrospettiva in cui il gruppo riflette sulle strategie adoperate, sulla scelta delle features/User Stories, dell'ordine del backlog, della differenza tra il costo di raffinamento e il costo di consegna valore ed, infine, su quali elementi portano o decrementano valore.

Carte Features

Il gioco è formato da carte features, presenti in due versioni, una originale e una raffinata.

Figura 3.5: Carte Feature [15]

A sinistra possiamo notare la carta originale con al centro riportata la sua dimensione (in questo caso S sta per small). A destra, invece, è riportata la carta raffinata che si ottiene in 4 User Stories, di cui sono riportati punti storia e valore per il cliente finale (nell'esempio, 14 story points per 1150 punti valore per l'utente finale).

Carte User Stories

Oltre alle carte features, Product Owner Value Game è composto anche dalle carte User Stories, anche queste presenti sia in versione originale che in versione modificata.

Figura 3.6: Carte User Stories [15]

Su 50 carte User Stories, solo 44 hanno due versioni (6 carte User Stories presentano solo la versione originale e non possono essere raffinate). A sinistra è presente la versione originale, che riporta al centro una dimensione espressa in story points (in questo caso 1) e in basso i Business Value Points bvp (nell'esempio 100). A destra vi è la carta raffinata (si può facilmente notare dal cerchio blu colorato) in cui sono descritti i bvp, in questo caso sono aumentati a 110, in quanto il lavoro ha portato all'incremento del valore per il cliente finale.

Capitolo 4

Analisi e confronto dei giochi proposti

Nel capitolo che segue, verrà eseguita una valutazione dei Serious Games precedentemente esaminati tramite l'approccio GQM (Game Model Quality). L'utilizzo di questo modello permette di analizzare in modo critico i giochi e di valutarne l'efficacia in termini di apprendimento, coinvolgimento e obiettivi formativi.
Come già visto, Escape the Boom, Scrumble e PO Value Game sono diventati strumenti sostanziali nell'ambito dell'educazione e della formazione. Tuttavia, è fondamentale comprendere se questi giochi riescano a raggiungere i risultati desiderati e se siano adeguati per il raggiungimento degli scopi formativi a cui sono destinati.
Nel corso di questo capitolo si andranno ad esplorare, quindi, i punti di forza e le aree di miglioramento di ciascun gioco, tenendo conto dei feedback dei partecipanti e dei dati acquisiti. Inoltre, verrà fatta una dettagliata considerazione su come questi giochi possano essere adattati e migliorati per massimizzare il loro impatto sull'apprendimento e sull'acquisizione delle competenze.

4.1 Approccio GQM

Per analizzare i risultati ottenuti, si è fatto uso del sistema GQM (Goal, Metric, Question), utile a creare linee guida orientate agli obiettivi nell'implementazione di un software. Questa metodologia di valutazione è stata creata originariamente per valutare i difetti di una serie di progetti presso la NASA Goddard Space Flight Center Environment [8]. Nonostante l'approccio venisse inizialmente utilizzato per creare e valutare gli scopi di un determinato progetto posto in un particolare ambiente, il suo uso è stato in seguito ampliato a vari contesti.

Tramite GQM, quindi, si impostano e poi analizzano i traguardi da raggiungere e vengono chiarite le domande di cui si cerca una risposta, ottenibile grazie all'insieme dei dati misurabili raccolti. Questo modello è strutturato su tre livelli:

- **Livello Concettuale (Goal):** si definsce un proposito per il team, in modo tale da poter soddisfare uno scopo specifico, ottenendo poi un risultato significativo.
- **Livello Operativo (Question):** viene creato un insieme di domande per valutare il raggiungimento degli obiettivi preposti.
- **Livello Quantitativo (Metric):** si associa ad ogni domanda un insieme di metriche, basate sul modello oggettivo. In questo modo è possibile raccogliere dei dati che siano misurabili in senso quantitativo.

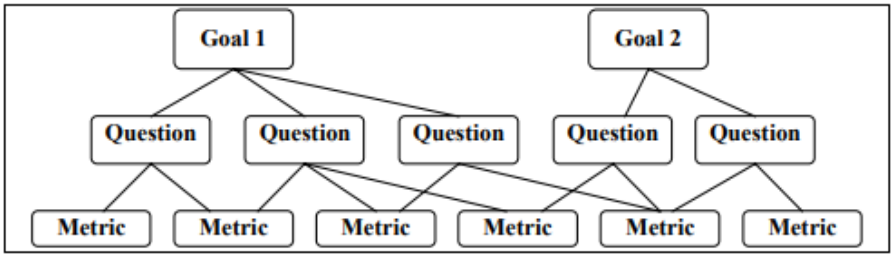

Figura 4.1: Struttura a livelli del metodo GQM [29]

L'approccio GQM si basa sul presupposto che, affinché un'organizzazione possa misurare utilmente i propri prodotti e processi, deve prima specificare gli obiettivi per sé stessa e per i propri progetti. Una volta fatto ciò, l'azienda deve essere in grado di ricondurli, utilizzando una gerarchia di domande correlate, ai dati destinati a valutare tali obiettivi a livello operativo. Infine, il metodo GQM consente all'organizzazione di ricavare un quadro (sempre tramite l'utilizzo delle domande) per interpretare i dati rispetto agli obiettivi dichiarati. Il risultato dell'applicazione dell'approccio GQM è la specifica di un sistema di misurazione mirato a un particolare insieme di questioni e un insieme di regole per interpretare i dati di misurazione. [35]
Quindi, un modello GQM è rappresentato da una struttura gerarchica che inizia con un obiettivo, che specifica lo scopo dell'analisi, l'oggetto da misurare, il problema preso in considerazione e il punto di vista da cui è tratto il calcolo. Viene poi raffinato l'obiettivo in diverse domande, solitamente dividendo il problema nelle sue componenti principali. Ogni domanda sarà in seguito migliorata, trasformandole in metriche (che possono essere sia oggettive che soggettive). Una stessa metrica può essere utilizzata per rispondere a domande diverse. Diversi modelli GQM possono anche avere domande e metriche comuni, facendo sì che, nel momento in cui la misura venga effettuata, i diversi punti di vista vengono presi in considerazione correttamente (ovvero, la metrica potrebbe avere valori diversi se presi da punti di vista diversi).

La metodologia GQM viene considerata dai professionisti utile sia per guidare iniziative di miglioramento che per sviluppare metriche per la gestione dei progetti software. Costituisce anche la base teorica per la progettazione di molti costrutti e strumenti utilizzati negli studi empirici di ingegneria del software. GQM è diventato un argomento fondamentale nei corsi avanzati di ingegneria del software, metodologia di ricerca e gestione di progetti software.
Per creare le tabelle di seguito riportate, si è fatto riperimento alla griglia di valutazione riportata a pag.17 dell'articolo *Cooperative Thinking: Analyzing a New Framework for Software Engineering Education* [9].

4.1.1 Valutazione GQM di una partita a Escape the Boom

Escape the Boom rappresenta un buon punto di partenza per l'educazione del Product Owner e costituisce una solida base per la comprensione del framework Scrum. Come visto in precedenza, infatti, questo gioco offre una simulazione di un ambiente in cui i partecipanti possono iniziare a sperimentare i principi e le dinamiche chiave di Scrum, ponendo un focus particolare sulla prospettiva del Product Owner.
Per questi motivi, è stato chiesto agli studenti del corso di Ingegneria del Software di effettuare una partita ad Escape the Boom prima di giocare a Scrumble, in modo tale da poter riscontrare eventuali differenze e/o preferenze tra i due giochi. Ad ogni gruppo è stata successivamente prosposta la tabella 4.2, in modo tale da permettere agli studenti di autovalutare la partita svolta.
Esaminando i risultati ottenuti, è emerso nitidamente come, secondo gli studenti, Escape the Boom sia caratterizzato da regole molto chiare e dinamiche, quindi utili ad accrescere la collaborazione e la partecipazione all'interno del gruppo, favorendo il team building. Inoltre, avendo all'interno del gioco dei ruoli attivi e ben definiti, il team ha lavorato con armonia per il raggiungimento di un obiettivo comune, in cui ogni membro era protagonista delle proprie scelte, portandoli a capire pienamente come il proprio impegno e lavoro possa portare ad una soluzione vincente. In più, dopo un'attenta analisi dei dati, si è potuto osservare come la maggior parte dei giocatori sia stata in grado di completare tutte le task proposte dal gioco entro una durata di tempo anche inferiore rispetto a quella stimata nella fase iniziale del gioco.
Infine si è notato come, tra i vari gruppi di studenti che hanno aderito all'esperienza, solo uno di essi non è riuscito a portare a termine la partita in quanto, alla conclusione di essa, si è riscontrato un valore di debito tecnico troppo alto, che eccedeva di molto gli intervalli preposti. Questo risultato evidenzia che Escape the Boom abbia favorito la comunicazione e la cooperazione continua all'interno del team e abbia fatto sviluppare nei membri del gruppo un senso di fiducia nelle figure dello Scrum Master e del Product Owner.

GOAL	QUESTION	METRIC	EVALUATION
Apprendimento	I membri del team hanno capito i ruoli di SCRUM?	Individuare i ruoli di SCRUM	**1** = non hanno capito i ruoli di Scrum **5** = conoscenza perfetta del proprio ruolo
	I membri del team hanno capito il processo?	Considerazioni del team	**1** = non si riesce a replicare il gioco **5** = può giocare come Scrum Master
	Hanno collaborato tutti?	Partecipazione dei singoli al livello	**1** = non capisce le dinamiche **5** = guida il gioco anche per gli altri membri
Practica	Le meccaniche del gioco sono chiare e ripetibili?	Opinioni dei partecipanti	**1** = crede che il gioco non possa essere replicato **5** = crede che il gioco possa essere eseguito anche in
	Il team ha completato il gioco con successo?	Numero di livelli completati	**1**= nessun livello superato **5** = tutti i livelli superati
	Il PO ha correttamente stimato prima dell'inizio del gioco?	Riconoscimento delle potenzialità del team	**1** = la stima è diversa dal risultato finale **5** = il risultato finale corrisponde alla stima
Cooperazione	I membri del team si conoscono meglio?	Livello di serenità tra i membri del team	**1** = non parla con gli altri giocatori **5** = parla con tutti
	Il gioco permette a tutti i giocatori di cooperare?	Livello di dialogo tra i membri del team	**1** = non si applica a collaborare **5** = sempre pronto a capire le dinamiche
	I membri del team si confrontano su i vari argomenti?	Scambio di idee durante il gioco	**1** = non ascolta le opinioni degli altri **5** = cerca argomenti per discutere
Motivazione	I membri del team si aiutano a vicenda?	Aiuto e collaborazione tra i giocatori	**1** = non è interessato al gioco **5** = controlla che tutti possano esprimersi
	Il PO aiuta il team?	Spiegazioni chiare e dettagliate del PO	**1** = spiegazioni confuse **5** = spiegazioni dettagliate e chiare
	Il team trova buone idee per il successo del gioco?	Livello completato con successo	**1** = non esprime pareri su come vincere **5** = le proprie idee portano ad una soluzione vincente
Problem Solving	Il team riesce ad affrontare eventuali problemi?	Tempo impiegato per finire il gioco	Se il numero di volte in cui la bomba è esplosa è > 4 allora mettere 1, se la bomba non è mai esplosa mettere
	Il team organizza i livelli correttamente?	Numero di livelli eseguiti con successo	Se per ogni livello la bomba è esplosa più di 2 volte mettere 1, se non è mai esplosa mettere 5
	Il PO programma correttamente il backlog?	Tempo impiegato per finire il livello	Stessa valutazione di **Q14**. ma fatta solo dal PO

Figura 4.2: Tabella GQM per la valutazione di Escape the Boom

Quindi possiamo concludere appurando che Escape the Boom fornisce un ambiente di apprendimento coinvolgente in cui i futuri Product Owner possono iniziare a sviluppare le competenze necessarie per avere successo nel loro ruolo. Questo gioco, infatti, non getta solamente le basi per la conoscenza di Scrum e dei ruoli che lo compongono, ma offre anche una piattaforma pratica per l'applicazione dei principi in situazioni reali, iniziando a formare i giocatori alla gestione di sfide più complesse presenti nella vita di tutti i giorni.

4.1.2 Valutazione GQM di una partita Scrumble

Successivamente, prima di iniziare ad implementare il software richiesto per il corso di Ingegneria del Software, è stato proposto agli studenti di effettuare una partita a Scrumble per comprendere nel dettaglio le componenti che formano il framework Scrum. Ad ogni team è stata, quindi, proposta la tabella 4.3 per autovalutare la partita di Scrumble effettuata.
Dall'analisi eseguita tramite le griglie valutative di tutti i team si è potuto notare come la partita a Scrumble sia stata uno strumento efficace ai vari gruppi che, grazie ad esso, sono riusciti ad avere un primo approccio al team building e alla collaborazione tra individui. Inoltre, si è notato come Scrumble abbia aiutato a creare le basi per lo sviluppo Agile, in quanto, durante lo svolgimento del gioco, sono stati definiti e compresi chiaramente i ruoli che si andranno a ricoprire all'interno del gruppo (Scrum Master e Product Owner).

Per quanto riguarda la parte relativa alla pratica, la maggior parte dei team è riuscita a portare a termine un numero di task considerevole e questo è stato possibile grazie ad una buona stima effettuata prima dell'inizio della partita. L'esecuzione di un numero elevato di task fa capire come i team siano riusciti a gestire e comprendere in modo efficace il lavoro da concludere, portando i giocatori a sviluppare abilità di pianificazione e organizzazione, elementi che risultano essere fondamentali in Scrum. Inoltre, esaminando i valori di debito tecnico ottenuti, si può osservare come questo risulti generalmente abbastanza basso, il che suggerisce che i giocatori hanno compreso l'importanza della qualità del lavoro e della gestione dei processi, in modo tale da riuscire a portare a termine le task in modo consapevole, evitando di accumulare problemi tecnici.

Infine, un risultato chiave emerso è l'efficace apprendimento dei ruoli in Scrum. Questo dimostra che il gioco ha fornito un'occasione importante per esplorare e comprendere i compiti e le responsabilità specifiche dei diversi membri del team, come il Product Owner, lo Scrum Master e i Developers.

GOAL	QUESTION	METRIC	EVALUATION
Learn	Do team members understand the Scrum roles?	Knowledge of Scrum roles by questions	1 = *no idea of the Scrum roles* 5 = *perfect knowledge of the roles and their jobs*
	Do team members feel they learned the process?	Opinions from the participants	1 = *couldn't repeat the game* 5 = *could play the game as a Scrum Master by himself*
	Does everyone keep up with the other players?	Check during every sprint retrospective if every one is on point	1 = *totally lost* 5 = *leads the game driving the other players*
Practice	Are the game mechanics linear and repeatable?	Opinions from the participants	1 = *feels the game is unrepeatable* 5 = *feels the game could be played in any situation*
	Do team success in completing the game?	Number of User Stories completed	1 = *0 to 3 stories* 2 = *4 to 6* 3 = *7 to 9* 4 = *10 to 12* 5 = *13 to 15*
	Do team members efficiently estimate during sprint planning?	Uniformity in evaluating the size and the priority of user stories	1 = *abnormal difference from the other players* 5 = *coherent and uniform with the group most of the time*
Cooperation	Do team members know each other better?	Level of players' serenity throughout the game	1 = *never speaks with the other players* 5 = *talks friendly to anyone in every situation*
	Does the game let all players cooperate?	Contribution of every player during the game	1 = *never puts effort in doing something* 5 = *every time is willing to understand what is going on*
	Do team member consult each other about a topic?	Sharing of ideas	1 = *never asks for an opinion* 5 = *wants to discuss about every topic*
Motivation	Do team members encourage collegues in need?	Players explain something other players don't understand	1 = *not involved by the game* 5 = *always makes sure everyone is on point*
	Does PO help the team?	Quality of PO's advices to get better in the next sprints	1 = *poor/absent advices* 5 = *wise and helpful suggestions when is required*
	Does the team come up with good ideas?	Effectiveness of sprint retrospective	1 = *doesn't express opinions during retrospective* 5 = *feels the retrospective fundamental to express opinions*
Problem Solving	Do team members behave well when facing a problem?	Level of the technical debt at the end of the game	*On the game board, if the debt pawn is on the lowest stage, the evaluation is* **5**, *for every higher stage it decreases by 1*
	Does team organize their tasks properly?	Average of tasks left at the end of each sprint	*Calculate the average of tasks left for each sprint:* 1 = *21+* 2 = *16-20* 3 = *11-15* 4 = *6-10* 5 = *0-5*
	Does PO plan efficiently the Sprint Backlog?	Average of tasks left at the end of each sprint	*Same evaluation as* **Q14** *for the PO*

Figura 4.3: Tabella GQM per la valutazione di Scrumble [6]

In generale, i dati favorevoli indicano che Scrumble ha contribuito in modo significativo alla comprensione di Scrum e all'applicazione efficace dei suoi principi. Il fatto che i partecipanti abbiano gestito le task in modo efficiente, limitando il debito tecnico e sviluppando una comprensione approfondita dei ruoli presenti nel framework Scrum

evidenzia l'utilità del gioco come strumento formativo. Questo risultato potrebbe tradursi in un'applicazione più efficace di Scrum nel contesto professionale, migliorando la gestione dei progetti e il conseguimento dei risultati desiderati.

4.1.3 Valutazione GQM di una partita a Product Owner Value Game

Infine, dopo aver completato le partite di Escape the Boom e di Scrumble, è stato chiesto agli studenti di fare un'ultima esperienza, ovvero quella di Product Owner Value Game. Tramite le autovalutazioni (Tabella 4.4) e le relazioni finali redatte dagli studenti, si è potuto notare come, mediante questo gioco, si è arrivati a comprendere appieno la figura e il ruolo del Product Owner e le scelte che è tenuto a compiere per cercare di arrivare a realizzare un prodotto che abbia valore massimo.
Grazie a questo gioco, il team ha dovuto cooperare per trovare delle idee che potessero portare all'ottenimento del maggior numero di bvp possibili. Infatti, come si è potuto notare dalle griglie valutative, il gruppo è riuscito a stabilire un forte legame con il Product Owner che ha tenuto conto di tutte le possibili opzioni di scelta suggerite dagli altri membri della squadra.

GOAL	QUESTION	METRIC	EVALUATION
Apprendimento	I membri del team hanno capito i compiti del PO?	Comprendere le scelte del PO	**1** = non hanno capito il ruoli del PO **5** = conoscenza perfetta del ruolo del PO
	I membri del team hanno capito il processo?	Considerazioni del team	**1** = non si riesce a replicare il gioco **5** = può giocare come PO
	Hanno collaborato tutti?	Partecipazione dei singoli al livello	**1** = non capisce le dinamiche **5** = guida il gioco anche per gli altri membri
Practica	Le meccaniche del gioco sono chiare e ripetibili?	Opinioni dei partecipanti	**1** = crede che il gioco non possa essere replicato **5** = crede che il gioco possa essere eseguito anche in altre situazioni
	Il team ha completato il gioco con successo?	Numero di sprint fatti	**1**= gioco finito al primo sprint **5** = eseguiti più di 3 sprint
Cooperazione e Motivazione	Il gioco permette a tutti i giocatori di cooperare?	Livello di dialogo tra i membri del team	**1** = non si applica a collaborare **5** = sempre pronto ad esprimersi
	Il team propone idee valide? **(SOLO PO)**	Scambio di idee durante il gioco	**1** = non ascolta le opinioni altrui **5** = cerca argomenti per discutere
	Il PO tiene conto delle idee degli altri giocatori? **(SOLO TEAM)**	Ascolto dell'opinione altrui	**1** = non esprime pareri su come vincere **5** = le proprie idee portano ad una soluzione vincente
Problem Solving	Il team riesce ad affrontare eventuali problemi?	Numero di story presenti nel backlog	**1** = nessuna story nel backlog **5** = più di 10 story nel backlog
	Il teamriesce ad arrivare ad una soluzione ottima?	Numero di carte feature raffinate	**1** = nessuna carta feature raffinata **5** = più di 7 carte feature raffinate
	Il team ha trovato una soluzione che massimizzi il valore?	Numero di bvp ottenuti	**1** = meno di 100 bvp ottenuti **5** = più di 800 bvp accumulati

Figura 4.4: Tabella GQM per la valutazione di Product Owner Value Game

Inoltre, andando ad analizzare la parte relativa al problem solving di ogni team, si può facilmente notare come la quasi totalità dei gruppi sia stata capace di affrontare eventuali difficoltà durante la partita, inserendo all'interno del Backlog un gran numero di User Stories e raffinando varie carte feature.
Infine, considerando il numero complessivo finale di Business Value Point (bvp) ottenuti alla fine della partita, si è notato come tutti i team siano riusciti ad averne una quantità superiore a 800, dimostrando, quindi, di essere stati in grado di selezionare le carte utili per sviluppare il prodotto più prestigioso.

Capitolo 5

Giochi di retrospettiva con Essence

Questo capitolo si apre con una panoramica del framework Essence, una metodologia strutturata volta a guidare i team agili durante il processo di sviluppo di un software. Essence è basato su principi e pratiche che puntano a migliorare l'efficienza e la qualità del progetto, promuovendo un approccio iterativo ed incrementale.
Successivamente, verranno descritte le carte proposte da Essence. Queste sono state progettate con lo scopo di facilitare la comunicazione e la collaborazione all'interno del team. Ogni carta rappresenta un elemento chiave del processo di sviluppo e viene utilizzata nel corso di sessioni di pianificazione o revisione per aiutare i membri del gruppo a discutere sugli aspetti critici in modo strutturato. Verrà poi spiegata l'importanza dell'adozione di Essence e delle sue carte nel contesto di redazione di una retrospettiva. Quest'ultima prevede degli incontri tenuti dal team per riflettere sul lavoro appena portato a termine, al fine di identificare perfezionamenti da apportare in futuro. L'uso del framework in questo contesto può migliorare notevolmente l'efficacia delle retrospettive, mettendo a disposizione un linguaggio condiviso e strumenti utili ad analizzare il processo di sviluppo.
Infine, verranno illustrati una serie di Serious Games da impiegare come esercizi pratici per iniziare a capire come condurre sessioni di retrospettiva produttive utilizzando il modello Essence. Attraverso l'uso dei giochi, i team possono sperimentare in modo interattivo e coinvolgente le pratiche e i principi chiave di Essence, acquisendo familiarità con le carte.

5.1 Framework Essence

Il framework Essence, reso uno *standard de facto* nel 2014 dall'Object Management Group (OMG) [16], nasce con lo scopo di dare una descrizione dettagliata degli elementi fondamentali che caratterizzano il mondo dell'Ingegneria del Software. Questo modello mira a fornire una base comune e un linguaggio condiviso per delineare gli step essenziali

dello sviluppo software, in modo tale che questi possano essere compresi, insegnati e applicati in modo coerente in diversi contesti.
Essence, modello realizzato per riprodurre le varie sfide che tutti i team di sviluppo software devono affrontare durante il processo, è stato implementato tramite un linguaggio visivo al fine di catturare le pratiche che aiutino i vari team ad affrontare tali sfide [19]. In questo sistema sono contenuti gli elementi su cui lavorare e focalizzarsi, le aree in cui sono necessarie attività di sviluppo e le competenze richieste per arrivare ad ottenere il successo.

Figura 5.1: Framework Essence: i sette alpha principali e le loro relazioni [19]

Il principio chiave su cui è basato il framework è il *kernel di Essence*, strumento utile a definire concetti importanti e che accomunano chiunque lavori in un determinato settore, come quello dell'Ingegneria del Software o dello sviluppo di un prodotto. Un kernel, quindi, è uno schema all'interno del quale molte pratiche diverse tra loro possono coesistere ed essere utilizzate con successo insieme. Esso include elementi essenziali e centrali in ogni attività di sviluppo software, tra cui:

- **Alphas:** usati per rappresentare i risultati desiderati o le entità fondamentali su cui un team di sviluppo concentra la propria attenzione per garantire il successo del progetto. Gli Alphas forniscono una struttura chiara per valutare lo stato di avanzamento di un progetto e identificare le aree che richiedono maggior attenzione. Inoltre, possono essere utilizzati come strumenti diagnostici per comprendere la salute del progetto e guidare il team nelle decisioni e azioni necessarie per raggiungere gli obiettivi preposti.

- **Spazi di attività:** descrivono gli ambienti in cui avvengono le attività durante il ciclo di vita del software. Ognuno di essi è focalizzato su un insieme specifico di

azioni correlate che contribuiscono al raggiungimento degli obiettivi del progetto. Gli Spazi Attività illustrano uno schema organizzativo chiaro che aiuta i team di sviluppo a identificare e gestire i compiti in modo coerente e orientato agli obiettivi. L'idea è quella di delineare una guida pratica su dove concentrare gli sforzi nelle diverse fasi del progetto, ottenendo così una gestione più consapevole e mirata delle attività di sviluppo.

- **Competenze:** fanno riferimento alle conoscenze, abilità e comportamenti che un team deve avere per portare a termine efficacemente le attività associate ad un determinato Alpha o Spazio Attività. Esse vengono generalmente suddivise in diverse categorie, ognuna correlata ad un aspetto chiave necessario per il successo del team. Le competenze forniscono un quadro pratico e generale utile ad identificare gli aspetti critici che devono essere migliorati e rafforzati all'interno del guppo.

- **Livelli di competenza:** indicano il grado di maturità e padronanza di un team rispetto ad una specifica competenza. Il concetto principale è che i gruppi progrediscano attraverso questi livelli nel tempo, ponendo come fine ultimo quello di raggiungere una conoscenza sempre maggiore. L'uso dei Livelli di Competenza nel framework Essence consente ai team di valutare in modo oggettivo il proprio progresso e lavoro, riuscendo ad identificare aree specifiche in cui è necessario un miglioramento. Ciò contribuisce a promuovere una crescita continua e a guidare il team verso una maggiore efficacia e successo nei progetti software.

Il modello Essence aiuta a strutturare il processo di riflessione e di analisi fornendo una serie di elementi chiave come i metodi di lavoro, gli obiettivi e le competenze, che possono essere degli strumenti usati per valutare lo stato attuale del progetto. Così facendo si facilita la discussione ed i membri del team vengono aiutati a concentrarsi sugli aspetti più rilevanti dei loro compiti. In più, viene favorito lo sviluppo di una visione olistica del processo: non bisogna limitarsi solo sugli aspetti tecnici, ma si devono includere anche considerazioni sul team, sui processi e sull'ambiente di lavoro. Seguendo quanto riportato nel libro "*The Essentials of Modern Software Engineering: Free the Practices from the Method Prisons!*" [20], infatti, è necessario riformare il modo in cui pensiamo al processo di sviluppo software, poichè bisogna liberarsi dai vincoli imposti da metodologie rigide e prescrittive per focalizzarsi su un approccio più flessibile e pragmatico, enfatizzando l'importanza delle pratiche rispetto alle strutture fisse. Questo concetto è in linea con quanto presente nell'approccio Agile, in quanto enfatizza l'importanza delle persone, della collaborazione e dell'adattabilità.
Inoltre, tramite l'uso del modello Essence, si riesce ad identificare e tracciare il progresso rispetto a degli obiettivi specifici. Avvelersi di esso durante le retrospettive aiuta i team a stabilire piani d'azione chiari e misurabili che portano ad avere un miglioramento nelle iterazioni future.

5.2 Carte Essence

La maggior parte dei team è abituata a rappresentare gli elementi di un progetto su delle carte, sia fisiche che elettroniche, in modo tale da utilizzarle come mezzo per organizzare i compiti da ultimare, stabilire le priorità e tener traccia del progresso. Le carte disposte dal framework Essence, invece, sono state create per raffigurare i concetti chiave alla base delle pratiche di lavoro, consentendo a tutti i gruppi di ragionare e migliorare notevolmente il proprio modo di cooperare.
Queste carte, piuttosto semplici e contenenti definizioni concise degli elementi essenziali di Scrum, sono state raggruppate in quattro categorie, ognuna delle quali rappresentata tramite un'icona.

- *Carte Alpha:* identificano gli elementi chiave da gestire in un progetto software. Vengono considerati degli indicatori delle prestazioni di un team in quanto sono completamente indipendenti dal processo e possono essere, quindi, applicati a qualsiasi contesto. Ogni Alpha, durante lo sviluppo, attraversa tre stati: programmato, pianificato e revisionato.

- *Carte Prodotto di lavoro:* raffigurano artefatti di valore e rilevanza per un team di sviluppo. Sono considerate di grande valore in quanto rappresentano gli elementi su cui investire e possono essere usate per facilitare la comunicazione di ciò che è stato discusso e concordato.

- *Carte Attività:* sono state progettate per guidare i team attraverso le varie fasi e mansioni del processo di sviluppo. Mentre le carte Alpha si concentrano sugli aspetti chiave o sugli elementi che bisogna progredire in un progetto, le carte Attività descrivono le azioni o i compiti che i membri del team devono eseguire per portare avanti questi elementi chiave verso il loro completamento.

- *Carte Pattern:* un pattern è un meccanismo generico utile a descrivere gli elementi e le relazioni presenti tra questi. Scrum fornisce due tipi di pattern, i principi, come un team cross-funzionale, e i ruoli, come, ad esempio, quello dello Scrum Master.

5.3 Retrospettive con carte Essence

L'utilizzo di Essence per redigere la retrospettiva di un Serious Game è utile ad approfondire la comprensione e l'applicazione dei principi Agili in un contesto ludico. Grazie ad esso si riescono a combinare elementi caratteristici di Scrum, attività utile a simulare le varie situazioni di sviluppo software in un ambiente controllato e di gioco, con l'approccio sistematico proposto da Essence per valutare e migliorare le pratiche di sviluppo.
Le carte fornite da Essence consentono di avere una struttura definita utile ad esplorare e conoscere i vari aspetti del lavoro del team, suddividendoli in categorie chiave come

Alphas, Attività e Pattern. Questo facilita il processo di valutazione e di analisi durante la sessione di retrospettiva. Inoltre, facendo uso del modello proposto da Essence, si incoraggia la partecipazione attiva di ogni membro del gruppo nello studio del proprio lavoro. Ogni carta mette a disposizione un punto di partenza da cui partire per avviare una discussione, consentendo a tutti di esprimere le proprie opinioni e di contribuire al processo decisionale.

Facendo riferimento a quanto riportato all'interno dell'articolo *"Better Scrum Through Essence"* [21], possiamo notare come il framework renda i vari compiti da portare a termine più chiari, semplici e facili da padroneggiare o da condividere, consentendo ai team di assumere pieno controllo sul loro modo di lavorare.
Usare il framework in corsi per l'educazione dei Product Owner ha mostrato come, mediamente, i team implementino correttamente solo un terzo delle componenti di Scrum, un terzo in modo scarso ed il restante non viene implementato affatto. Per questo motivo, grazie all'uso di giochi e attività che favoriscono l'auto-riflessione e l'apprendimento, l'adozione di Essence può portare ad un miglioramento significativo dell'efficacia con cui i gruppi implementano le task di Scrum [21].

5.3.1 Team Status Game

Il serious game Team Status Game rappresenta il primo approccio al modello Essence. Questo gioco, infatti, è basato sull'uso delle carte fornite da Essence per condurre retrospettive efficaci, valutando collettivamente lo stato di avanzamento del team.
Durante la partita, ogni membro del gruppo sceglie una carta che rappresenta lo stato attuale del team. Se tutti i singoli individuano la stessa carta allora si può passare allo strato successivo, altrimenti si avvia una discussione dove ognuno espone le proprio motivazioni sulla scelta appena compiuta.
Una volta che si è raggiunta l'unanimità sulla carta, si va ad osservare la checklist presente in essa; in segreto, ogni partecipante deve selezionare un numero (inferiore al numero di elementi presenti nella checkbox della carta). Si scoprono, quindi, i numeri e, cominciando da chi ha quello più basso, ogni membro del gruppo indica quali compiti, tra quelli presenti nella carta, sono stati portati a termine.
Si prosegue finchè tutti hanno 0 voti. A questo punto, se ogni voce nella checkbox è stata valutata, significa che il team è già nello stadio successivo, altrimenti bisogna controllare cosa non è stato spuntato, avviando una riflessione per progettare un piano d'azione.

Team Status Game stimola la riflessione condivisa, portando il team a valutare le proprie prestazioni e promuovendo una comprensione condivisa degli obiettivi di miglioramento, rendendo il processo di retrospettiva più coinvolgente e produttivo.

5.3.2 Alpha State Cards Game

Il gioco Alpha State Card Game è stato progettato con lo scopo di guidare il team attraverso una valutazione strutturata dei vari aspetti del processo di sviluppo software rappresentati dalle carte Alpha, identificando le aree di forza e le opportunità di miglioramento.

Lo Scrum Master da inizio alla partita in quanto ha il compito di identificare delle carte rilevanti per il gruppo. Quindi, i membri del team scelgono, tra le carte selezionate, quelle che raffigurano gli aspetti critici del processo. Si avvia, in seguito, una discussione in cui vengono individuate le aree che necessitano di maggiore attenzione e quelle che, invece, rappresentano un punto di forza per il gruppo. A questo punto, il team sviluppa un piano d'azione che include attività specifiche, responsabilità assegnate e scadenze, al fine di garantire il progresso e il monitoraggio delle azioni di miglioramento.

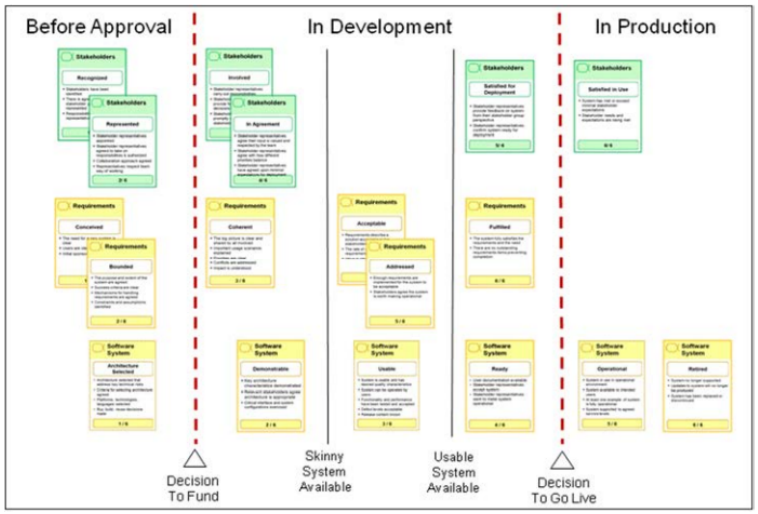

Figura 5.2: Alpha State Cards Game

Essence fornisce una base solida per la strutturazione e la conduzione di Alpha State Card Game, rendendo la valutazione dello stato del team più chiara e focalizzata. Le carte Alpha offrono una rappresentazione chiara degli elementi chiave da prendere in considerazione e le checklist associate aiutano ad avere una guida dettagliata utile all'analisi.
Inoltre, Essence favorisce la standardizzazione e la condivisione delle pratiche, permettendo al gruppo di adottare un linguaggio comune per discutere le prestazioni. Questo porta ad ottenere una migliore comprensione e collaborazione tra i membri del team, facilitando una valutazione più accurata e approfondita dello stato attuale e la definizione di azioni di miglioramento mirate.

5.3.3 Scrum

Un concetto fondamentale di Scrum è la retrospettiva, ovvero il momento in cui il team esegue un'analisi su ciò che è successo durante lo sprint, in modo tale da poter creare un piano per perfezionare le proprie prestazioni. Questa può essere fatta utilizzando le carte Essence, uno strumento utile ad identificare e dare priorità a questi miglioramenti. Le carte vengono spostate, ordinate o raggruppate durante sessioni collaborative durante le quali il team prende decisioni o riflette sul proprio modo di cooperare.
Viene creato un tabellone bidimensionale in cui le colonne rappresentano quanto sia importante quel concetto per il gruppo, mentre le righe fanno riferimento a quanto bene la squadra sente di essersi comportata in quel determinato ambito. Il team, quindi, posiziona le carte sulla tavola in modo appropriato e le muove mentre discute e ne analizza il funzionamento [21].
Le carte in alto a destra sono quelle che richiedono maggiore attenzione in quanto considerate importanti e migliorabili (nel caso in cui ce ne fossero molte, si può assegnare una priorità, scegliendo su quali di esse focalizzarsi). Infine, i membri del gruppo approfondiscono la natura dei problemi e vengono proposte delle attività che possono essere usate nell'iterazione successiva.

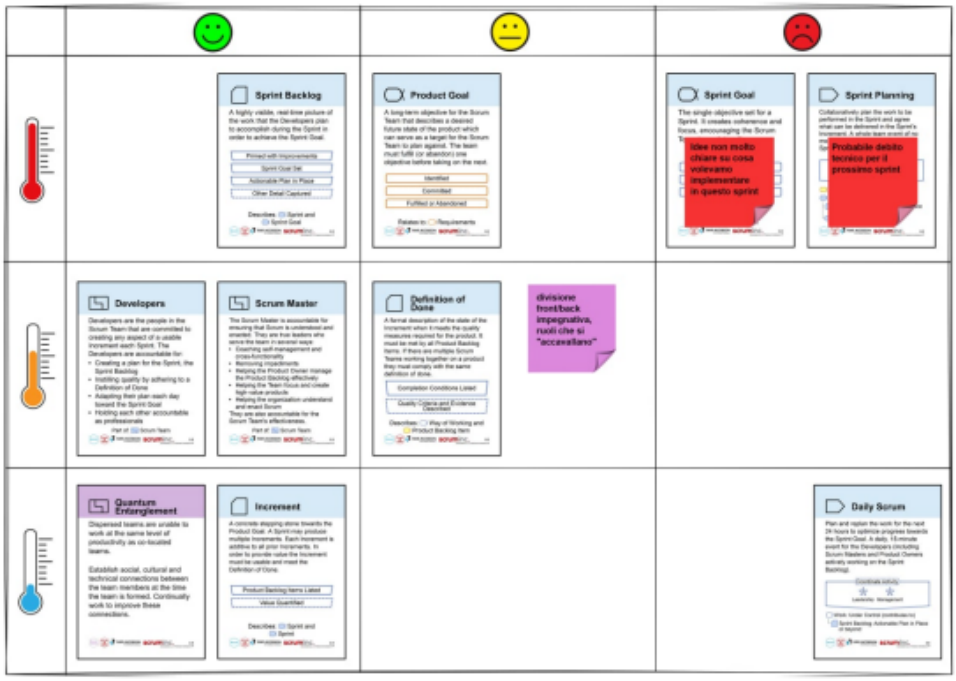

Figura 5.3: Retrospettiva con carte Essence

Il contenuto delle carte riassume brevemente le informazioni più importanti contenute all'interno della guida di Scrum. Questo aiuta ad evitare eventuali discussioni o conflitti, dovuti ad interpretazioni personali di un concetto, da parte dei singoli membri del gruppo. Organizzare in questo modo la retrospettiva permette ai nuovi team di avere un primo approccio al mondo Scrum, aiutandoli a capire i diversi meccanismi che lo caratterizzano. Inoltre, spinge il gruppo ad agire e a pianificare nel dettaglio i lavori che possono portare ad un miglioramento concreto.

Capitolo 6

Conclusioni

Il costante sviluppo presente nel mondo moderno ha portato le figure del produttore e del consumatore ad unirsi più che mai. Questo è il frutto dell'aumento delle discipline di programmazione iterativa che interessano i team interfunzionali seguaci del modello Agile.
Al giorno d'oggi, accresce sempre di più il numero di organizzazioni che si stanno adoperando per passare da ambienti tradizionali (basati sul modello "a cascata"), in cui il valore viene aggiunto alla fine di ogni rilascio, a contesti improntati sulle metodologie Agile/Scrum, cambiando un piano "fail-safe" per uno "safe to fail", focalizzandosi maggiormente sull'apprendimento e miglioramento ad ogni iterazione. L'elemento chiave di Scrum è un team di piccole dimensioni, interfunzionale e auto gestito, in cui figurano tre ruoli fondamentali: lo Scrum Master, i Developers e il Product Owner.
Facendo riferimento a quanto riportato dai creatori del framework Scrum, il Product Owner è la figura responsabile della massimizzazione del valore del prodotto e del lavoro del team di sviluppo [33]. Quindi, per far arrivare questa persona al successo, è necessario che l'intero team rispetti le sue decisioni, maturate grazie ad una efficace gestione del Product Backlog e tramite l'impostazione di obiettivi chiari e raggiungibili, adattati alle esigenze del gruppo e dei clienti. Pertanto, il Product Owner rappresenta una figura centrale e critica ed è considerato il nesso tra tutti i nodi. Allo stesso tempo, però, risulta particolarmente complesso il processo che porta a diventare esperti in questo ambito, poiché il ruolo stesso è fortemente influenzato dall'ambiente lavorativo in cui è posto. [23]
Le responsibilità del Product Owner variano da un'organizzazione all'altra, per cui la sua implementazione è raro che sia conforme a quanto riportato nella guida ufficiale di Scrum.

Nel corso di questo elaborato, si sono dapprima analizzati i ruoli chiave presenti all'interno di Scrum, per focalizzarsi poi sulla figura del Product Owner e di come essa abbia bisogno di una formazione solida ed efficace. L'uso di giochi educativi, quali i Serious Games, si è dimostrato un approccio altamente vantaggioso, utile a comprendere al meglio

i principi alla base del framework Scrum e allenando i futuri Product Owner a guidare con successo lo sviluppo del prodotto, massimizzando il valore per il cliente.
Per analizzare quanto detto, facendo riferimento ai giochi presi in esame, si può comporre un percorso educativo strutturato in tre fasi:

- **Fase 1**: si inizia con il gioco *Escape the Boom* che risulta essere uno strumento utile a gettare le basi per un primo approccio alla teoria del team building. I partecipanti all'esperienza vengono suddivisi in piccoli gruppi, ognuno formato da individui con diverse capacità e competenze. Questa ripartizione riflette la diversità presente all'interno di team di lavoro reali e porta i partecipanti ad affrontare la prima sfida, ovvero quella di imparare a lavorare insieme.

  Il gioco, seppur in maniera divertente, stabilisce un obiettivo comune al gruppo, quello di disinnescare un ordigno. La riuscita di questa missione incoraggia la cooperazione e la collaborazione. Infatti, i partecipanti devono imparare a comunicare tra loro in maniera chiara ed esaustiva, ascoltando le opinioni altrui ed esprimendo le proprie idee. Inoltre, la partita spesso è caratterizzata da vincoli di tempo o da risorse limitate, situazioni simili a quelle presenti nel mondo reale. Questo può essere usato come un primo allenamento per gestire le risorse in modo ottimale e a prendere decisioni rapide.

- **Fase 2**: dopo un primo impatto con le dinamiche presenti all'interno di un team di sviluppo, ci si può addentrare nel mondo di *Scrumble*, un gioco che riassume al meglio le caratteristiche fondamentali del framework Scrum. Durante la partita, i giocatori devono collaborare per gestire efficacemente il backlog del prodotto. Questo compito include la definizione delle proprietà e delle funzionalità, la stima del carico di lavoro e l'adeguata allocazione delle risorse. Inoltre, Scrumble porta i team a pianificare e governare gli sprint, suddividendo le attività e assegnando compiti specifici. Questa breve esperienza porta i giocatori a comprendere l'importanza di una corretta pianificazione e dell'organizzazione che vi è nel mondo Scrum.

  Il gioco offre situazioni che simulano la complessità e le difficoltà che possono sorgere nel corso dello sviluppo di un software. Grazie a ciò, i partecipanti imparano a controllare i rischi e ad adattarsi alle sfide del mondo reale.

- **Fase 3**: infine, l'ultima tappa del percoso formativo del Product Owner è rappresentata da *Product Owner Value Game* in quanto offre ai giocatori l'opportunità di mettere in pratica le conoscenze acquisite con i giochi precedenti, ricoprendo il ruolo del Product Owner. Nel corso della partita, i partecipanti si assumono le responsabilità del ruolo ricoperto, andando a definire le varie priorità dei compiti e scegliendo quali di essi portare a termine per massimizzare il valore del prodotto da consegnare al cliente.

Tramite questo gioco si possono simulare le pressioni e le sfide che il Product Owner spesso deve affrontare nella vita reale, come ad esempio il cambio improvviso dei requisiti o i vari limiti di tempo. Inoltre, grazie a PO Value Game, si enfatizza l'importanza di massimizzare il valore per il cliente, ovvero uno degli aspetti fondamentali che stanno alla base del ruolo di Product Owner.

In conclusione, questo lavoro mostra come la formazione di un Product Owner possa essere perfezionata tramite l'uso di Serious Games, fornendo un approccio pratico e coinvolgente, che riflette i principi alla base della metodologia Agile. La combinazione di apprendimento esperienziale, acquisizione della pratica attraverso delle situazioni simulate e una maggiore consapevolezza del valore da produrre è un mezzo utile ad aiutare i futuri Product Owner ad ottenere successo nel loro ruolo.

Nonostante ogni azienda possa personalizzare questo ruolo, tenendo però conto del contesto lavorativo in cui è collocato, si deve sempre considerare che lo svolgimento effettivo di questa funzione dipende molto dalle caratteristiche presenti in ogni individuo. È particolarmente importante la capacità della persona di consolidare e mantenere relazioni all'interno del team, e questo compito richiede delle buone capacità di comunicazione.

Bibliografia

[1] Clark C Abt. *Serious games*. University press of America, 1987.

[2] J Arafeen and Saugata Bose. Improving software development using scrum model by analyzing up and down movements on the sprint burn down chart: Proposition for better alternatives. *International Journal of Digital Content Technology and its Applications*, 2009.

[3] Emily Bache. Pipeline - the game that delivers!, 2019.

[4] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development, 2001.

[5] Mike Beedle, Martine Devos, Yonat Sharon, Ken Schwaber, and Jeff Sutherland. Scrum: An extension pattern language for hyperproductive software development. *Pattern languages of program design*, 1999.

[6] Roberto Borbone. Virtual team building through serious games: the scrumble case study. Master's thesis, University of Bologna, 2020.

[7] Luigi Buglione and Alain Abran. Improving the user story agile technique using the INVEST criteria. In *Proc. Joint 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*. IEEE, 2013.

[8] Victor R Basili1 Gianluigi Caldiera and H Dieter Rombach. The goal question metric approach. In *Encyclopedia of software engineering*, pages 528–532. 1994.

[9] Paolo Ciancarini, Marcello Missiroli, and Daniel Russo. Cooperative thinking: Analyzing a new framework for software engineering education. *Journal of Systems and Software*, 157:110401, 2019.

[10] Mike Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[11] Melvin E Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.

[12] Giuseppe Costanza. Cosa sono le user story e come si scrivono, 2020.

[13] Michael Cramer and Achim Stremplat. Bomb defuser field manaual, 2015.

[14] Michael Cramer, María Berenguer, and Marjoke Franken. Team building and remote collaboration with "escape the boom". *ProjektMagazin*, 2020.

[15] Paul Kuijten Dajo Breddels. Product owner value game, 2015.

[16] Object Management Group. Omg essence standard kernel and language for software engineering methods. *Kernel and Language for Software Engineering Methods (Essence), v1.2*, 2018.

[17] J Richard Hackman and Ruth Wageman. A theory of team coaching. *Academy of management review*, 30(2):269–287, 2005.

[18] Alice Hoon, Emily Oliver, Kasia Szpakowska, and Philip Newton. Use of the 'stop, start, continue'method is associated with the production of constructive qualitative feedback by students in higher education. *Assessment & Evaluation in Higher Education*, 40(5):755–767, 2015.

[19] Ivar Jacobson International. Essence explained. https://www.ivarjacobson.com/essence-explained-agile-tools, 2024.

[20] Ivar Jacobson, Pan-Wei Ng, Paul E McMahon, Michael Goedicke, et al. *The essentials of modern software engineering: free the practices from the method prisons!* ACM & Morgan-Claypool, 2019.

[21] Ivar Jacobson, Jeff Sutherland, Brian Kerr, and Barbora Buhnova. Better Scrum through Essence. *Software: Practice and Experience*, 52(6):1531–1540, 2022.

[22] Alsever Jennifer, Hempel Jessi, Taylor III Alex, and Roberts Daniel. 6 great teams that take care of business. *Fortune, April*, 10, 2014.

[23] Maja Due Kadenic, Diego Augusto de Jesus Pacheco, Konstantinos Koumaditis, Gitte Tjørnehøj, and Torben Tambo. Investigating the role of Product Owner in Scrum teams: Differentiation between organisational and individual impacts and opportunities. *Journal of Systems and Software*, 206:111841, 2023. ISSN 0164-1212.

[24] Allan Kelly. *The Art of Agile Product Ownership*. Springer, 2019.

[25] Jeanine Krath, Linda Schürmann, and Harald FO Von Korflesch. Revealing the theoretical basis of gamification: A systematic review and analysis of theory in research on gamification, serious games and game-based learning. *Computers in Human Behavior*, 125:106963, 2021.

[26] Maria Lencastre, Daniel Silva, João Henrique C Pimentel, and Jaelson Brelaz Castro. PRIUS: Applying Gamification to User Stories Prioritization. *SIGAPP Applied Computing Review*, 23(4):27–44, 2024.

[27] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. The use and effectiveness of user stories in practice. In Maya Daneva and Oscar Pastor, editors, *Requirements Engineering: Foundation for Software Quality*, pages 205–222. Springer, 2016.

[28] Elton Mayo. The hawthorne experiment. western electric company. *2016). Classics of organization theory*, pages 134–141, 1933.

[29] Harekrishna Misra. Relevance of measurements in e-governance: Software engineering perspectives in indian context, 07 2015.

[30] Sreeram Mullankandy. Agile metrics — what matters and why? *Agile Insider*, 2019.

[31] Adrien Muller and Yoan Thirion. Yata - a devops game, 2017.

[32] Maria Paasivaara, Ville Heikkilä, Casper Lassenius, and Towo Toivola. Teaching students scrum using LEGO blocks. In *Companion Proceedings of the 36th International Conference on Software Engineering*, 2014.

[33] Ken Schwaber and Jeff Sutherland. Scrum. *URL: http://www. scrumalliance. org/system/resource/file/275/whatIsScrum. pdf,[Sta nd: 03.03. 2008]*, 2010.

[34] Ken Schwaber and Jeff Sutherland. The scrum guide. *Scrum Alliance*, 2011.

[35] F. Shull, C. Seaman, and M. Zelkowltz. Victor r. basili's contributions to software quality. *IEEE Software*, 23(1):16–18, 2006. doi: 10.1109/MS.2006.33.

[36] Ana Silva, Thalles Araújo, João Nunes, Mirko Perkusich, Ednaldo Dilorenzo, Hyggo Almeida, and Angelo Perkusich. A systematic review on the use of definition of done on agile software development projects. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 2017.

[37] Pyxis Technologies. Scrumble - an agile board game to discover and improve your skills as a scrum team!, 2015.

[38] Yagulawathi Thangarajoo and A Smith. Lean thinking: An overview. *Industrial Engineering & Management*, 2015.

[39] Eva Tsahuridu and Gillian Vesty. Transforming accounting ethics education with serious games. https://shorturl.at/cnNPV.

[40] Bruce W Tuckman. Developmental sequence in small groups. *Psychological bulletin*, 63(6):384, 1965.

[41] Gerard Wagenaar, Remko Helms, Daniela Damian, and Sjaak Brinkkemper. Artefacts in agile software development. *Springer*, 2015.

[42] Dave West. Agile Scrum roles and responsibilities. https://www.atlassian.com/agile/scrum/roles.

[43] Tracey Williams. Role of a Project Manager in an Agile World, 2022.