Politecnico di Torino

Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Gamification Applicata al GUI Testing di Applicazioni Web

Relatore:
Prof. Luca Ardito

Candidato:
Cacciotto Filippo

Co-relatore:
Dott. Riccardo Coppola

Anno Accademico 2020/21
Torino

Sommario

Contesto: Il GUI Testing (Graphical User Interface Testing) di applicazioni web è una tecnica di System Testing basata sull'interazione con l'interfaccia grafica dell'applicazione da testare, eseguita compiendo le stesse azioni che metterebbe in atto l'utente finale durante la navigazione del sito. Nonostante l'efficacia di questa pratica di validazione, spesso i tester tendono a trascurare questo step fondamentale dello sviluppo di software per via dei suoi costi elevati e della quasi totale assenza di feedback immediati riguardanti la qualità delle proprie sessioni di testing.

Obiettivo: Questa tesi propone l'applicazione di alcuni concetti di Gamification al GUI Testing, ossia l'inserimento di elementi e meccanismi tipici del game desing all'interno di contesti non ludici, in questo caso nell'esecuzione delle sessioni di testing, al fine di incentivare i tester a migliorare la qualità delle proprie performance.

Metodo: La tesi si concentra sulla teorizzazione e l'applicazione di cinque elementi di Gamification, implementati all'interno di un plugin Java inserito nel contesto di un prototipo di software di GUI Testing: Scout.

Risultati: La versione finale del software di GUI Testing, comprensiva del plugin di Gamification, è stata sottoposta ad una fase di validazione eseguita da un piccolo campione composto da studenti e lavoratori, tutti laureati (o laureandi) in Informatica o Ingegneria Informatica.

Conclusioni: L'introduzione della Gamification ha portato ad un incremento della coverage media del 47%, oltre che ad un complessivo miglioramento dell'esperienza complessiva in termini di motivazione e coinvolgimento.

Indice

Elenco delle figure 5

Elenco delle tabelle 7

1 Background e stato dell'arte 9
- 1.1 Introduzione . . . 9
- 1.2 Gamification . . . 10
  - 1.2.1 Teorie sull'applicazione della Gamification . . . 10
  - 1.2.2 Applicazioni nell'istruzione . . . 17
  - 1.2.3 Applicazioni nell'Ingegneria del Software . . . 19
  - 1.2.4 Applicazioni nel software testing . . . 24
  - 1.2.5 Limiti . . . 29
- 1.3 GUI testing e tool esistenti . . . 31

2 Architettura e design 35
- 2.1 Scout . . . 35
- 2.2 Descrizione dei file di log . . . 38
- 2.3 Rappresentazione della sessione . . . 39
- 2.4 Estrapolazione delle metriche . . . 42
- 2.5 Adattamento del tool . . . 44
- 2.6 Gamification Engine . . . 45
- 2.7 Elementi di Gamification applicati . . . 49
  - 2.7.1 Barra di progresso . . . 49
  - 2.7.2 Punteggio . . . 51
  - 2.7.3 Classifica . . . 57
  - 2.7.4 Easter Egg . . . 58
  - 2.7.5 Evidenziazione delle pagine scoperte . . . 60

2.7.6 Elementi teorizzati 61

3 Esempio di sessione 65

4 Validazione del plugin 69
4.1 Selezione del campione 69
4.2 Metodologia 71
4.2.1 Descrizione dell'esperimento 71
4.2.2 Preparazione dell'ambiente di svolgimento 75
4.3 Analisi delle performance dei tester 76
4.3.1 Coverage media 76
4.3.2 Punteggio 77
4.3.3 Easter Egg 79
4.4 Risultati del questionario 80
4.4.1 Barra di progresso 81
4.4.2 Stella 82
4.4.3 Punteggio 83
4.4.4 Classifica 84
4.4.5 Easter egg 85
4.4.6 Giudizio complessivo 86

5 Limiti di applicabilità 91
5.0.1 Limiti di validità interna 91
5.0.2 Limiti di validità esterna 93

6 Conclusioni 97
Riferimenti bibliografici 100

Elenco delle figure

1.1 Fasi del framework. 11

2.1 Panoramica del tool. 36
2.2 State Model. 37
2.3 Package contenente le classi che permettono di rappresentare una sessione all'interno del plugin. 41
2.4 Package contenente le classi che implementano le funzionalità relative agli elementi di Gamification. 46
2.5 Barra di progresso di una sessione di testing nella pagina di Wikipedia. 50
2.6 Barra di progresso di una sessione di testing nella pagina di Wikipedia. 51
2.7 Cattura della schermata di riepilogo, contenente tutte le metriche relative alla sessione appena conclusa. 56
2.8 Cattura della schermata contenente lo storico relativo ad un tester. 57
2.9 Cattura della schermata contenente la classifica dei dieci tester che hanno totalizzato il maggior numero di punti nel corso delle varie sessioni di testing. 58
2.10 Cattura della schermata in cui è presente l'easter egg rappresentato come un ovale giallo. 59
2.11 Una pagina mai stata visitata, contrassegnata dalla stella gialla in trasparenza. 60

3.1 E-shop del sito scelto per l'esempio. 66
3.2 Rappresentazione grafica dell'albero della sessione. 67
3.3 Output della visita pre-order dell'albero della sessione. 68

4.1 Anni di esperienza dei partecipanti nella programmazione web. . . . 70
4.2 Linguaggi utilizzati dai partecipanti per la programmazione web. . . 70

4.3 Anni di esperienza dei partecipanti nel testing di applicazioni web. . 71
4.4 Organizzazione dell'esperimento. 72
4.5 Coverage media delle sessioni a confronto. 77
4.6 Punteggio ottenuto dai tester nelle due sessioni. 78
4.7 Easter Egg scoperti dai tester nelle due sessioni. 79
4.8 Percentuale di easter egg a confronto. 80
4.9 Valutazione dell'utilità della barra di progresso nel mostrare i progressi parziali. 82
4.10 Valutazione dell'efficacia della stella nell'incoraggiare esplorazioni più approfondite. 83
4.11 Valutazione dell'impatto motivazionale della presenza di un punteggio finale. 84
4.12 Valutazione della comprensibilità delle informazioni contenute nella schermata di riepilogo. 84
4.13 Valutazione dell'efficacia della classifica nello stimolare il senso di sfida dei tester. 85
4.14 Valutazione dell'efficacia degli easter egg nell'invogliare i tester ad interagire con il maggior numero di widget all'interno di ogni pagina. 86
4.15 Numero di partecipanti convinti dalla presenza di uno specifico elemento ludico. 87
4.16 Valutazione della facilità di integrare il tool in un contesto di testing esistente. 87
4.17 Valutazione della capacità degli elementi ludici di rendere i tester consapevoli dei propri progressi e dei propri risultati. 88
4.18 Valutazione sulla preferibilità della versione ludicizzata di Scout rispetto a quella originale. 88

Elenco delle tabelle

1.1 Template di un'Attività 13

2.1 Voti basati sul punteggio finale 53

4.1 Domande del Questionario 73
4.1 Domande del Questionario 74
4.2 Suddivisione partecipanti 76

Capitolo 1

Background e stato dell'arte

1.1 Introduzione

Nello sviluppo di un software, il testing costituisce un'attività fondamentale, arrivando ad incidere anche per oltre il 40% del costo totale. Nonostante l'importanza di tale attività, spesso le aziende finiscono per lesinare, dedicando un quantitativo di tempo e di risorse molto inferiore rispetto a quello che essa richiederebbe normalmente. Svolgere la mansione di testing, inoltre, è spesso considerata una pratica noiosa e poco stimolante.

Testare in maniera inadeguata il software in produzione espone gli sviluppatori a rischi molto elevati: è infatti molto frequente che, in seguito alle lacune descritte in precedenza, il software prodotto possa essere affetto da problemi di sicurezza, comportamenti indesiderati e disservizi che vanno a ledere l'esperienza vissuta dall'utente finale. Posticipare la risoluzione dei problemi successivamente al rilascio del software, inoltre, risulta essere molto più costoso che farlo in seguito ad una accurata fase di testing atta appositamente a rivelare i malfunzionamenti nel codice.

In questo studio ci si concentrerà sul GUI Testing, ossia un tipo di software testing che si focalizza sugli aspetti legati all'interfaccia grafica, facendo controlli sulle schermate e tutti gli elementi che le compongono, come i bottoni, le aree di testo, le icone e tutti gli altri elementi visivi. Questa categoria di test è particolarmente importante nel contesto delle applicazioni web e delle app mobile, in cui le interfacce grafiche sono in continua evoluzione e possono cambiare release dopo release, con piccole modifiche o, addirittura, stravolgimenti completi.

Questo lavoro, in particolare, tratterà lo sviluppo di un plugin che introduca

delle meccaniche di Gamification ad un software usato per testare l'interfaccia grafica di applicazioni web (Scout), al fine di stimolare i tester nelle loro mansioni. Di seguito verrà analizzato il tema della Gamification da diverse prospettive e verranno mostrati gli studi precedenti che sono stati analizzati e sulla base dei quali si è costruito il plugin.

1.2 Gamification

La **Gamification** (o *ludicizzazione*) consiste nell'uso degli elementi, della filosofia e delle meccaniche tipiche del game design in contesti non ludici (Deterding et al., 2011). Esistono diversi approcci per applicare gli elementi tipici dei giochi al fine di supportare obiettivi diversi dal consueto intrattenimento: per esempio, in diversi contesti vengono utilizzati i cosiddetti *serious games*, ossia giochi veri e propri che hanno però finalità tipicamente educative. La Gamification, invece, si differenzia da questi ultimi in quanto un sistema ludicizzato non è un gioco in tutto e per tutto, ma uno strumento che ha altre finalità e che integra al suo interno delle meccaniche di gioco per cercare di ottenere determinati benefici. Un sistema ludicizzato, quindi, si compone andando ad integrare un ambiente di Gamification nel contesto dell'attività principale.

L'idea alla base del concetto, quindi, è quella di sfruttare tutte quelle caratteristiche che rendono i giochi attraenti e divertenti (al punto tale da creare dipendenza in certi casi), al fine di migliorare l'esperienza di un certo target di utenti in ambienti non ludici come il posto di lavoro, la scuola o l'università, ossia tutti contesti dove lo scopo principale non è il divertimento.

1.2.1 Teorie sull'applicazione della Gamification

Molti studi sulla Gamification si concentrano sui risultati ottenuti dall'applicazione di tali tecniche in vari ambiti e sui benefici che esse possono apportare, mentre solo una piccola parte affronta il problema dal punto di vista del design di un sistema ludicizzato. Nonostante tutti i benefici citati in precedenza, infatti, le applicazioni ludicizzate non sono semplici da implementare, poiché si deve tenere conto di molti fattori per evitare di realizzare un sistema scadente e fallimentare.

Un modo per identificare e tentare di risolvere alcuni eventuali problemi fin dalla fase di progetto di un sistema ludicizzato consiste nella creazione (o nell'utilizzo)

di un framework metodologico per l'integrazione della Gamification nei processi di sviluppo di software, come quello proposto in (Herranz et al., 2014) e (Herranz et al., 2013) e rappresentato in Figura 1.1.

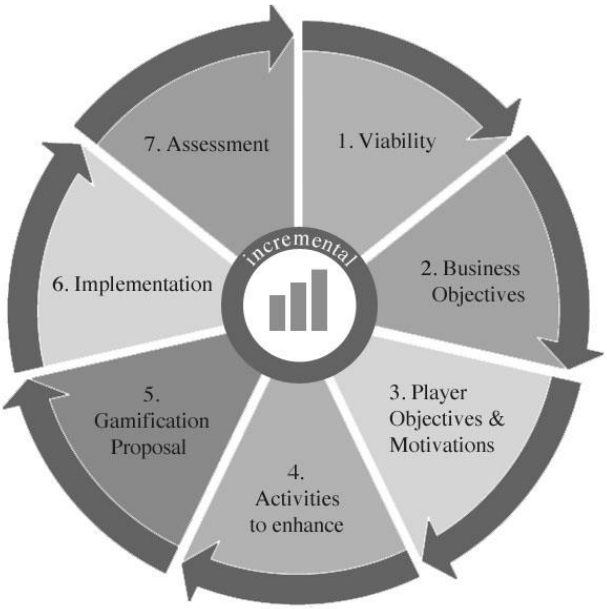

Figura 1.1. Fasi del framework.

Il framework si suddivide di 7 fasi:

1. *Viability*: questa fase valuta la fattibilità dell'implementazione della Gamification nell'organizzazione in esame;

2. *Business Objectives*: si fa un'ulteriore valutazione riguardo alla possibilità di implementazione della Gamification sulla base degli obiettivi del progetto;

3. *Player Objectives and Motivations*: si esplorano le motivazioni dei vari gruppi professionali che lavoreranno al progetto;

4. *Activities to Enhance*: si identificano le attività del processo migliorabili grazie alla Gamification e si discute sulle strategie da adottare per farlo;

5. *Gamification Proposal*: il fulcro del framework, la fase in cui si propone un'effettiva implementazione del sistema ludicizzato;

6. *Implementation*: effettiva realizzazione e integrazione della proposta della fase 5;

7. *Assessment*: si analizzano i risultati ottenuti e gli obiettivi raggiunti.

Un altro framework per la ludicizzazione dell'Ingegneria del Software è stato presentato da (Dal Sasso et al., 2017) ed è rappresentato in Tabella 1.1. Esso si basa sul concetto di **Attività**, a sua volta composta dai concetti di **Analisi**, **Implementazione** e **Testing**. Ogni attività riguarda uno specifico tipo di utente (*ruolo*) presente nei sistemi ludicizzati, tra cui troviamo:

- *Observer*: agiscono in modalità di sola-lettura;
- *Writer*: interagiscono modificando il contenuto già presente;
- *Solver*: realizzano gli obiettivi del sistema ludicizzato.

Normalmente le persone che interagiscono col sistema passano dinamicamente da un ruolo all'altro.

Come detto in precedenza, il concetto fondamentale del framework è l'**attività**, che è composta da un ID, una breve descrizione e una lista degli *elementi costitutivi*, ossia "bulding block" che riprendono degli elementi tipici della Gamification, più pertinenti. Ogni attività viene a sua volta strutturata in tre fasi:

1. **Analisi**: ogni attività all'interno dell'ambiente ludicizzato deve basarsi su un *fondamento logico* facilmente comprensibile da collegare agli obiettivi globali dell'ambiente, oltre che su un *obiettivo emotivo*, che rappresenta i benefici che si vogliono ottenere nelle persone che interagiscono con il sistema.

2. **Implementazione**: per implementare un'attività bisogna innanzitutto avere una chiara idea degli *attori* in gioco, oltre che delle *dinamiche* in cui verranno coinvolti, che rappresentano sostanzialmente le tattiche da adottare per coinvolgere le persone in una certa attività. Tutto ciò viene istanziato utilizzando componenti tipici del game design chiamati *meta*, come livelli, vite e classifiche. Infine, è necessario riflettere sui *rischi* che possono sorgere dalla struttura ludica scelta (ad esempio problemi algoritmici, comportamenti indesiderati, requisiti dell'hardware ecc.).

3. **Testing**: fase che consiste nel capire quali siano i *target* dell'attività di testing, quale *metodologia* possa essere utilizzata per eseguire i test e, infine, quali siano i *risultati attesi*, da confrontare con quelli *reali*, per agevolare un approccio iterativo di sviluppo del sistema ludicizzato.

Attività	
Ruolo & ID	
Descrizione	
Elementi Costitutivi	
Analisi	Fondamento Logico Obiettivo Emotivo
Implementazione	Attori Dinamiche Meta Rischi
Testing	Target Metodologia Risultati Attesi Risultati Reali

Tabella 1.1. Template di un'Attività

Un'attività dipende da uno o più **elementi costitutivi**. Ne esistono dieci tipi:

1. *Portale*: consiste in una ricompensa (ad esempio un badge) di benvenuto, assegnata ad un utente non appena attraversa i confini della piattaforma, registrandosi e fornendo le informazioni necessarie per farsi identificare all'interno della community.

2. *Produzione*: elemento che ha lo scopo di rendere immediatamente produttivo un utente sulla piattaforma, per evitare che ci sia un calo di interesse. Si può suddividere in tre sotto-elementi:

    - *Simbiosi*: si esegue un'attività che aiuta direttamente o indirettamente un altro utente;
    - *Narciso*: si esegue un'azione per migliorare la propria posizione all'interno della community, così da aiutare l'utente a capirne i meccanismi;

- *Alveare*: si deve proporre un'idea per migliorare la piattaforma e la vita della community.

3. *Coraggio*: è importante premiare in maniera equa gli utenti che riescono ad eseguire determinati task, specialmente i più complicati. Aumentare la fiducia in sé stessi degli utenti potrebbe stimolarli a voler provare task più difficili, il cui completamento potrebbe accrescere le loro competenze e quindi renderli in grado di affrontare attività progressivamente più complicate.

4. *Mischia*: questo elemento fa riferimento alla mischia del rugby per porre enfasi sull'importanza della forza del gruppo. Collaborare, condividere risorse utili, competere e socializzare con altri membri della community è intrinsecamente motivante, quindi il sistema deve premiare e promuovere il lavoro di squadra e la competizione.

5. *Camaleonte*: elemento che fa riferimento alla dinamicità con cui devono mutare gli achievement proposti dal sistema. Supponendo che l'utente, utilizzando il sistema, acquisisca sempre più competenze che lo spingeranno a cercare sfide sempre nuove (*Coraggio*), l'ambiente dovrebbe reagire a questi cambiamenti introducendo nuovi achievement e rilasciando ricompense ad-hoc, che entreranno poi a far parte della libreria di sistema così da poter essere ottenuti anche da altri utenti. Allo stesso modo, achievement che non vengono raggiunti da nessun utente per troppo tempo dovrebbero essere eliminati dalla libreria per via della loro probabile impraticabilità.

6. *Fulmine*: una volta diventato esperto, l'utente potrebbe annoiarsi e si potrebbe assistere ad un calo della sua produttività. E' in questa situazione che l'ambiente dovrebbe "colpire l'utente come un fulmine", dirigendolo verso una nuova sfida particolarmente motivante con premi personalizzati.

7. *Phasing*: gli utenti nel mondo virtuale possono compiere delle azioni che hanno un impatto permanente sull'ambiente circostante. Questo elemento suggerisce di mutare la piattaforma in relazione alla progressione della competenza di ciascun utente. Di conseguenza, due utenti in diverse fasi della propria progressione vedranno rappresentazioni differenti dello stesso scenario e potranno interagire con esso in modi diversi.

8. *Abbellimento*: è importante che l'aspetto degli avatar degli utenti cambi nel tempo, diventando sempre più attraente man mano che si progredisce nell'ambiente virtuale. Al contrario, in caso di inattività, deve degradarsi sempre di più col passare del tempo.

9. *Champagne*: dato che gli achievement virtuali sono importanti per gli utenti, bisogna permettere loro di celebrare i propri successi virtuali anche nel mondo reale, per esempio permettendo di inserire all'interno del proprio CV, un link diretto al proprio profilo virtuale.

10. *Ascensione*: normalmente i giochi hanno una fine, ma nel contesto di un processo produttivo è importante non porre un limite del genere alle attività per evitare che alcuni utenti diventino inattivi. E' bene però premiare nel mondo reale gli utenti che hanno ottenuto risultati particolarmente positivi all'interno dell'ambiente virtuale.

Oltre ai framework appena descritti, sono stati realizzati numerosi altri studi considerabili come punto di partenza per il design di un sistema ludicizzato. Tra i più significativi, troviamo analisi effettuate dal punto di vista della figura del project manager, volte a studiare come applicare le tecniche di Gamification ad un progetto.

Nello studio di (Platonova & Bērziša, 2017) viene mostrato come i benefici della Gamification (aumento della motivazione, riduzione dello stress e intrattenimento durante un'attività lavorativa) siano alcuni degli obiettivi ricercati da un project manager per il proprio team di lavoro.

Secondo le autrici, gli elementi di Gamification più comuni e di maggior successo sono basati su punteggi e classifiche: l'efficacia di tali elementi è dovuta al fatto che viene dato un feedback immediato al giocatore. L'assegnazione dei punteggi ai task da completare deve seguire un criterio proporzionale che può tuttavia anche condurre al fenomeno della *pointsification*. Questo comportamento indesiderato verrà discusso più approfonditamente in seguito.

La classifica è un altro elemento importante in quanto inserisce una componente di rivalità e di sfida tra le persone coinvolte nell'ambiente ludicizzato. I badge e gli achievement, invece, sono meccaniche meno utilizzate secondo le autrici poiché meno motivanti in quanto serve molta pazienza per ottenerli.

Secondo (Dal Sasso et al., 2017), quando ci si approccia al design di un ambiente ludicizzato si devono considerare diversi aspetti, ponendosi le seguenti domande:

- *Determinazione*: da dove scaturisce la motivazione per incoraggiare un certo comportamento?
- *Scelte significative*: le attività scelte per la ludicizzazione sono abbastanza significative?
- *Struttura*: si può spiegare con un algoritmo il comportamento desiderato?
- *Potenziale conflitto*: Il gioco evita la tensione con un'altra struttura emotiva (gioia, competitività, ecc.)?

Inoltre, la Gamification deve avere lo scopo di espandere le interazioni già esistenti piuttosto che quello di rendere semplicemente divertente un compito noioso: questo perché talvolta si rischia di ottenere un effetto contrario (repulsione per una Gamification imposta o fenomeno della *pointsification*).

Infine, nello studio viene proposta una serie di indici per valutare l'efficacia dell'impatto del layer di Gamification, tre dei quali vengono forniti dal punto di vista tecnico, mentre due dal punto di vista emotivo: *success metrics, analytics, conflitti, jen ratio, questionari.*

- *Success Metrics*: quantità definite per ogni attività; devono essere specifiche per ciascuna piattaforma, in modo che si riferiscano a quantità significative e facili da misurare.
- *Analytics*: sono metriche che misurano l'interazione degli utenti con la piattaforma (dati come il *Daily Active User* servono per contare gli utenti unici di un certo giorno, mentre il rapporto *Daily/Monthly Active User* fornisce un trend sull'uso del tool in un certo momento).
- *Conflitti*: sono i possibili errori che sorgono dal momento dell'introduzione del layer di Gamification. Tali errori devono essere al più presto identificati e risolti, altrimenti va rimosso il layer stesso.
- *Jen Ratio*: valuta quanto sia positivo l'atteggiamento degli utenti, misurando le interazioni positive e quelle negative (rapporto tra le interazioni positive

totali tra gli utenti rispetto alle interazioni negative totali in un dato periodo di tempo e contesto). Il risultato è compreso tra 0 e 1: più è prossimo a 1, migliore è il benessere percepito dalla community.

- *Questionari*: vengono somministrati periodicamente a utenti di diversi livelli di competenze, ponendo domande non solo tecniche, riguardanti il funzionamento del sistema, ma anche sullo stato emotivo derivante dall'uso del sistema stesso.

È stato infine evidenziato come un buon indice di predizione per un sistema ludicizzato sia il fatto di poter conoscere anticipatamente se i giocatori, ossia gli utilizzatori del sistema, giochino regolarmente nel loro tempo libero oppure no (Mollick & Rothbard, 2014): una persona già abituata avrà meno difficoltà a comprendere le meccaniche di gioco e ci sono maggiori probabilità che possa mostrare interesse.

1.2.2 Applicazioni nell'istruzione

Nel campo dell'istruzione, la Gamification viene utilizzata principlamente per aumentare la motivazione ed il coinvolgimento, oltre che per migliorare le performance degli studenti durante i corsi. Di seguito verranno presentati alcuni esempi di applicazione delle meccaniche ludiche in ambito accademico.

Nel loro studio, (Anderson P. & R., 2015) descrivono e analizzano un tentativo di ludicizzazione di un corso di data science tramite **Learn2Mine**, un sistema di apprendimento open-source cloud-based che sfrutta la Gamification per formare data scientist in maniera divertente e coinvolgente.

Il sistema è basato su un meccanismo di scomposizione di un problema più complicato in diversi sottoproblemi e step intermedi (facoltativi) proposti in maniera incrementale. L'idea è quella di guidare lo studente nella scrittura del codice richiesto partendo dalla realizzazione delle sue singole parti (spesso indipendenti le une dalle altre, eccezion fatta per lo step finale che dipende da tutti i precedenti).

Learn2Mine sfrutta principalmente barre dell'esperienza, punti, badge e classifiche per stimolare la competizione fra gli studenti. Il sistema fornisce inoltre un feedback immediato sull'attività svolta dallo studente ed elargisce ricompense e achievement in base ai risultati raggiunti. Tutti questi elementi sono integrati in

maniera non intrusiva, permettendo a chiunque non fosse interessato agli aspetti legati alla Gamification di ignorarli con facilità.

Learn2Mine, inoltre, grazie al meccanismo di feedback immediato, solleva gli insegnanti dall'incarico di dover fornire assistenza continua agli studenti, permettendo loro di avere una panoramica sui progressi raggiunti nell'apprendimento. I docenti hanno inoltre la possibilità di distribuire manualmente delle ricompense agli studenti per i loro risultati parziali o di modificare i voti (e/o i punteggi) che non ritengono coerenti con le linee guida stabilite all'inizio del corso.

Al termine del corso oggetto dello studio, sonos state raccolte le valutazioni degli studenti riguardo al sistema tramite un questionario basato su scala Likert e la media delle valutazioni è risultata concorde sull'efficacia del sistema. In particolare, i vantaggi principali sono stati individuati nella possibilità di ritentare più volte di risolvere uno stesso problema e nei feedback immediati forniti dal sistema. Learn2Mine, inoltre, ha aumentato la percentuale di consegna delle soluzioni portandola dal 40% all'80-90% di media nel giro di un solo anno di utilizzo. Secondo gli autori dello studio, Learn2Mine ha portato anche un miglioramento sia nella comprensione che nell'apprendimento degli argomenti proposti.

Gli studenti hanno mosso però anche delle lamentele verso alcune funzionalità del sistema: certe riguardavano il fatto che la scomposizione dei problemi in sottoproblemi più piccoli fosse eccessiva, altre il fatto che gli step intermedi fossero tutti indipendenti dagli altri, altre ancora il fatto che gli errori segnalati dal sistema fossero poco chiari.

Un altro tentativo di applicazione della Gamification in ambito accademico è stato effettuato da (Buckley & Clarke, 2018), che hanno proposto l'erogazione di un corso di *Software Testing* in due modalità, allo scopo di metterle a confronto.

Le modalità di insegnamento sono: *SEP-CyLE* (Software Engineering and Programming Cyberlearning Environment), che include collaborative learning, social interaction e problem-based learning, e *WReSTT-CyLE* (Web-Based Repository of Software Testing Tutorials: a Cyberlearning Environment), che al contrario non include gli elementi di problem-based learning, chat e ratings ed è più simile ad una raccolta di tutorial riguardanti gli argomenti delle varie lezioni.

La Gamification è stata applicata attribuendo dei punti nei seguenti modi: quando si completa un Digital Learning Object, superando i quiz, interagendo sul forum

e completando le attività. È presente anche un parte di interazione sociale realizzata tramite la creazione di un profilo per lo studente, l'invio di messaggi nel forum per la comunicazione e il rating di post e Digital Learning Object. Inoltre, troviamo anche una classifica che mostra gli avatar dei 3 studenti che hanno ottenuto più punti.

Uno studio più recente ha invece provato ad applicare la Gamification in modo differente: anziché focalizzarsi sulla metodologia di erogazione della didattica come nei casi descritti in precedenza, si integrano le meccaniche ludiche nella valutazione del codice prodotto degli studenti durante il corso (Zsigmond et al., 2020).

Il sistema proposto ha lo scopo di insegnare concetti di programmazione usando come elementi di Gamification i punti e i badge. Viene utilizzata una piattaforma web in cui ogni studente deve consegnare il proprio codice entro determinate scadenze, stabilite anticipatamente per ogni assignment. Il sistema analizza il codice presentato prima in modo statico, controllandone la correttezza sintattica, e poi dinamicamente, mandandolo in esecuzione in una sandbox e ridirigendo input e output in modo controllato: per questo controllo dinamico vengono usati un numero predefinito di test cases dipendenti dall'assignment.

I badge hanno un prerequisito di sblocco noto a tutti gli studenti che, quando soddisfatto, certifica di aver raggiunto certi obiettivi: esso ha lo scopo di suscitare l'interesse e l'impegno dei partecipanti a produrre codice funzionante e ad applicare i concetti appresi. Vi sono due tipi di badge:

- *Di stile*: incoraggiano gli studenti all'adozione di pratiche di programmazione corrette e a cimentarsi nello sviluppo di codice in tal senso;

- *Achievement*: certificano le performance dello studente e forniscono inoltre dei punti bonus validi per l'esame finale, che spingono gli studenti a competere tra loro in quanto alcuni badge possono essere assegnati solo ad un numero limitato di persone.

1.2.3 Applicazioni nell'Ingegneria del Software

La Gamification merita una menzione particolare per quanto riguarda la sua applicazione nel campo dell'Ingegneria del Software, poiché i processi di sviluppo di

software sono fortemente human-intensive e ciò la rende uno strumento promettente in grado di aumentare il coinvolgimento quotidiano e la motivazione degli sviluppatori.

Nello studio realizzato da (Passos E.B., 2011) viene descritto un tentativo di Gamification del processo di sviluppo di un software. L'idea di base nasce non solo dal tentativo di sfruttare i benefici che la ludicizzazione potrebbe potenzialmente apportare (quali maggior divertimento e coinvolgimento degli sviluppatori), ma anche dall'idea di vedere lo sviluppo di un software come un susseguirsi di task (*missioni*) durante i quali si devono acquisire delle competenze per poter superare sempre nuove sfide, ottenendo ricompense o penalizzazioni a seconda del risultato.

Lo studio si concentra molto sul game design, ponendo particolare enfasi sull'organizzazione e la gerarchia dei livelli (disposti in un grafo orientato contenente le sfide) e lo skillset necessario per superarli. Sono inoltre presenti altre caratteristiche:

- L'*esito* di una sfida è modellato con una funzione che ritorna 1 in caso di successo, 0 in caso di fallimento, prendendo in input le variabili che compongono lo stato del gioco al momento della chiamata. Il risultato di una sfida dipende dal risultato ottenuto dal giocatore nelle n sottosfide che compongono quella principale.

- Gli *achievement*, assegnati in due modi: sulla base della *repetition*, in base a quante volte uno stesso tipo di sfida è stato superato e assegnati basandosi su una scala logaritmica che rende i livelli successivi via via più difficili da superare, oppure sulla base del *rate*, la percentuale di successi rispetto ai tentativi in un certo intervallo di tempo, assegnati basandosi su una scala lineare e sul superamento di determinate soglie. Gli achievement vengono distribuiti sulla base di diverse metriche (per esempio il code coverage nel caso del testing) sia individuali che collettive (relative ai progetti).

- *Feedback* immediato, al quale viene data molta importanza, in quanto viene ritenuto un aspetto fondamentale per rendere il giocatore consapevole dei propri progressi.

Andando a mappare tutti questi concetti di game design con le fasi tipiche dell'Ingegneria del Software, le sfide equivalgono di fatto alle release di un software

e ognuna di esse si può scomporre in una serie di iterazioni che costituiscono le sottosfide o i task. I nodi foglia del grafo dei livelli sono task immediati.

E' importante calibrare attentamente la difficoltà delle sfide per renderle interessanti, evitando che siano troppo facili o troppo difficili così da non incorrere nel fenomeno del *burnout*, ossia la situazione in cui una skill viene padroneggiata ad un livello tale da renderne noioso l'utilizzo. Un buon modo per evitare il burnout è quello di usare espedienti come lo storytelling, in modo da distrarre il giocatore e mantenerlo interessato al gioco.

Il sistema analizzato nello studio sfrutta elementi tipici dei *Role-Playing Games* (noti anche come *RPG* o *giochi di ruolo*), come punti esperienza (assegnabili anche in numero minore se un task non viene eseguito entro una certa scadenza, per esempio), attributi del personaggio (i cui punti vengono assegnati automaticamente in base alle sfide superate e non scelti dal giocatore) e classi (anch'esse automaticamente attribuite).

Gli autori dello studio hanno notato che l'uso degli achievement non ha soltanto incrementato il coinvolgimento degli sviluppatori, ma ha anche permesso di monitorare e controllare il processo di sviluppo nel suo complesso. Si pone inoltre l'accento sulla possibilità di creare un legame fra uno sviluppatore e il suo avatar nel gioco, cosa che può portarlo a svolgere con più interesse le proprie attività quotidiane. Questo aspetto è legato al fatto che, diversamente da quanto accade in un videogioco, le abilità del personaggio e i suoi progressi sono strettamente legati al programmatore in quanto riflettono i suoi progressi nel mondo reale. Tutti questi aspetti hanno portato gli autori a valutare positivamente l'integrazione del layer di Gamification nel processo di sviluppo.

Un altro esempio di applicazione della Gamification all'Ingegneria del Software, questa volta incentrato sul suo insegnamento piuttosto che su un effettivo processo di sviluppo, viene descritto da (Berkling & Thomas, 2013). Anche in questo caso, il principale motivo dell'adozione di un sistema di ludicizzazione è legato ai benefici che questo tipo di approccio riesce spesso ad apportare in termini di partecipazione e motivazione dei soggetti coinvolti. In particolare, il corso di Ingegneria del Software analizzato nello studio è stato organizzato proprio concentrandosi sulle teorie psicologiche riguardanti la motivazione: secondo diversi studi, infatti, gli esseri umani sono motivati a lavorare su task difficili dal punto di vista cognitivo quando vengono loro garantite autonomia (*autonomy*), uno scopo (*purpose*) e la

possibilità di padroneggiare ciò che stanno facendo (*mastery*). Il corso ha tenuto quindi conto di questi 3 aspetti:

- è stato seguito un percorso chiaro con varie possibilità in termini di ordine e velocità di svolgimento delle attività (*autonomy*);
- è stato stabilito un obiettivo ben definito e sperabilmente gratificante (*purpose*);
- gli studenti hanno avuto la possibilità di sostenere più volte alcuni esami durante il corso (*mastery*).

Tutti questi concetti sono stati presi in considerazione nell'integrazione della Gamification, applicata inserendo elementi quali barre di progresso, classifiche e ricompense. È inoltre possibile vedere i progressi degli altri studenti e mandar loro messaggi, così da incentivare non solo la competizione ma anche la collaborazione (premiata con ricompense speciali).

L'idea di base è quella di organizzare gli argomenti del corso su una mappa divisa in diverse aree, ognuna rappresentante un diverso topic e ognuna suddivisa in 3 parti. L'obiettivo di uno studente è quello di acquisire un certo insieme di conoscenze completando almeno due parti su tre di una determinata area, così da sbloccare l'accesso al *marketplace* di quell'area e di conseguenza la possibilità di creare un gruppo di colleghi per poter iniziare un progetto. All'interno di questi *marketplace* è possibile pubblicizzare il proprio gruppo o unirsi ad uno di quelli esistenti. È inoltre presente una chat con cui gli studenti possono chiedere aiuto o fornire assistenza ai propri colleghi, con la possibilità di ottenere dei punti bonus.

L'esperimento non aveva ottenuto i risultati sperati, ma gli studenti avevano trovato interessanti gli elementi legati alla *task overview*, ossia la possibilità di monitorare i progressi raggiunti nelle proprie attività, e gli elementi legati al *feedback*.

Un'applicazione della Gamification ad un aspetto dell'Ingegneria del Software piuttosto inusuale, ossia la definizione delle specifiche formali, è quella studiata da (Prasetya et al., 2019) con **FormalZ**. Da quello che ci è dato sapere, questo è l'unico tentativo di usare tecniche di ludicizzazione per insegnare a livello accademico i concetti relativi a questa fase dello sviluppo del software.

Gli aspetti ludici sono molto ben studiati e integrati: viene raccontata la storia di un hacker che cerca di compromettere una CPU e il giocatore ha come scopo

quello di proteggerla dall'attacco. Per attuare delle protezioni, lo studente deve scrivere del codice per implementare un filtro efficace per contrastare gli attacchi , per poi usarlo come torre di difesa della CPU.

Il tool segue la meccanica dei *tower defense game* per imparare a scrivere le specifiche formali delle varie funzioni Java, mascherando l'attività con un'impalcatura molto allettante: costruire un filtro per proteggere la CPU sotto attacco. Lo studente deve ricostruire la formula corretta assemblando dei blocchetti (ognuno dei quali può rappresentare una variabile, una costante o un operatore) usando i fili per assemblare i blocchi in modo da ricostruire correttamente la formula.

Il sistema riconosce se la specifica sia equivalente a quella fornita dal professore, facendo una verifica semantica della formula scritta ed eventualmente, a discrezione dell'insegnante, anche un test random: si assegnano gli stessi valori casuali alle espressioni del professore e dello studente controllandone nuovamente l'equivalenza.

Questo tool è stato testato solamente in un modo preliminare su un corso di *Software Testing and Verification* dando comunque buoni risultati per quanto riguarda il coinvolgimento degli studenti. Gli aspetti che non hanno convinto i soggetti del test sono principalmente legati alla comprensione delle meccaniche di gioco, alcune delle quali non sono risultate abbastanza chiare sin dal principio (può essere dovuto al tutorial troppo breve o al design troppo complesso).

Citiamo infine una delle prime applicazioni della Gamification allo sviluppo del software che riguarda tutte le fasi dello sviluppo: **HALO** (Highly Addictive, sociaLly Optimized), analizzato nello studio di (Sheth et al., 2011).

HALO è uno strumento utilizzato per inserire una componente caratteristica dei giochi *MMORPG* (Massively Multipayer Online Role Play Game) all'interno degli IDE più comuni tramite l'installazione di un apposito plugin. Questo plugin utilizza un elemento peculiare ossia le *quest*, atte a mascherare i task di tutti i giorni dietro ad un "sipario ludico", utilizzando anche un sistema di ricompense per i compiti portati a termine che comprende punti esperienza, valuta virtuale, titoli e ricompense reali.

Le quest comprendono task che hanno uno spettro di difficoltà molto variabile: ne esistono alcune che comprendono un solo compito introduttivo di complessità minima, altre che comprendono molti incarichi completabili da una sola persona e quest che invece coinvolgono necessariamente più persone per essere portate a

compimento. Inoltre, esse possono essere divise in sub-quest completabili indipendentemente oppure legate tra loro da un ordine particolare. Le sub-quest possono inoltre essere obbligatorie per il completamento della quest principale oppure opzionali, lasciando al giocatore la libertà di portarle a compimento per ottenere un maggior punteggio o una ricompensa extra.

Gli elementi ludici presenti hanno diverse funzioni:

- i *punteggi* servono semplicemente ad indicare l'esperienza del giocatore e il suo livello;
- la *valuta virtuale* può essere spesa per fare acquisti all'interno della piattaforma per personalizzare il proprio alter ego;
- i *titoli* sono soprannomi esclusivi che si possono attribuire al proprio alter ego solo dopo aver conseguito un traguardo particolare (ad esempio "The Bug Slayer" si potrebbe sbloccare dopo aver fatto report e fix di un numero elevato di bug);
- le *ricompense reali* sono degli omaggi che si possono ottenere in cambio di un numero elevato di monete virtuali o al conseguimento di particolari eventi (alcuni esempi possono essere una gift card in cambio di un grosso numero di monete virtuali o ancora un pranzo offerto dall'azienda per tutto il team che ha raggiunto un traguardo importante come il porting dell'applicazione da un sistema opertivo ad un altro).

1.2.4 Applicazioni nel software testing

Come già detto in precedenza, una delle fasi più dispendiose dello sviluppo di un software in termini di tempo e di risorse è quella di testing. Nonostante sia cruciale avere una test suite completa ed esaustiva per poter riconoscere e prevenire le problematiche derivanti da bug sparsi nel codice, spesso il software viene testato in maniera blanda per poter rispettare le scadenze imposte, trascorrendo maggior tempo nella fase di scrittura del codice. Inoltre, la fase di testing è considerata spesso noiosa e ripetitiva e non in grado di stimolare l'attenzione del tester.

Per rendere più piacevole tale disciplina sono stati fatti vari tentativi, tra cui l'applicazione della Gamification. Vediamo di seguito alcuni esempi.

Un primo esempio di tool che applica la Gamification ad un'attività di testing è **Code Defenders**, ideato da (Rojas & Fraser, 2016). Questo strumento ludicizza l'attività di mutation testing rendondola interattiva e divertente. Il linguaggio predefinito è Java, ma è possibile estendere il tool all'utilizzo di altri linguaggi.

Il sistema definisce due ruoli: gli *attaccanti*, che devono introdurre dei bug nel codice per mettere in luce le fragilità della test suite esistente, e i *difensori*, che devono invece migliorare la test suite arricchendola di test supplementari al fine di renderla più robusta. L'attaccante, apportando delle modifiche al codice, crea un *mutante*: se un mutante riesce a superare tutti i test della test suite originale, ma fallisce dei test nella nuova test suite arricchita dal difensore, allora il difensore ottiene un dterminato punteggio. L'attaccante, viceversa, ottiene dei punti se il mutante creato non è stato rintracciato con nessun "fail" sulla test suite arricchita. Il punteggio assegnato dipende dal numero di test che un mutante riesce a superare.

Alcuni mutanti sono sintatticamente differenti ma semanticamente uguali al codice originale da testare: questi mutanti sono detti *equivalenti* e non sono considerati validi nel sistema ludicizzato. Se il difensore sospetta che l'attaccante abbia prodotto un mutante equivalente invece che uno normale può invocare un *duello*. Se l'attaccante è sfidato a duello, deve dimostrare a sua volta di aver prodotto un mutante non equivalente creando una test suite nuova con dei test che facciano fallire tale mutante. Se l'attaccante non è in grado di produrre tale test suite, allora il sistema considera il mutante equivalente e tutti i punti raccolti dall'attaccante sono persi.

Code Defenders ha un'implementazione web-based e può essere giocato sia in coppia che in gruppo. È anche possibile giocare in modalità singola, ma necessita l'integrazione di un tool di generazione dei test e di un tool di generazione automatica dei bug.

Code Defenders è stato impiegato come strumento per insegnare software testing a diversi livelli accademici con esiti molto incoraggianti (Fraser et al., 2019). Inoltre è emerso come la test suite risultante sia molto più robusta rispetto a quelle generate automaticamente coi tool esistenti.

L'introduzione di questa piattaforma in un corso universitario deve essere graduale, per permettere agli studenti di familiarizzare con l'ambiente prima di approfondire gli argomenti. Le classi Java per le quali si devono creare test suite e mutanti devono essere scelte dal professore in modo che abbiano una complessità crescente ma sempre bilanciata: sbilanciare la test suite o la classe in esame può

favorire una delle due parti che si sfidano.

Le classi Java usate nello studio sono state prima una introduttiva creata da zero, poi due classi matematiche con metodi noti agli studenti, mentre successivamente si è passati ad altre classi esistenti nel mondo open-source. Particolarmente interessanti sono le classi che rappresentano strutture dati in quanto dotate di poche dipendenze di codice o completamente prive.

Le coppie sono state formate in modo da essere le più bilanciate possibile: gli studenti che hanno ottenuto un punteggio simile nella stessa sessione laboratorio vengono accoppiati nel laboratorio successivo. Sono state anche svolte sessioni in cui non ci si esercita a coppie, ma in gruppi di attacco contro gruppi di difesa, alcune delle quali sono state eseguite con numeri non bilanciati, basandosi sulla "forza" degli studenti (studenti più deboli raggruppati a difendere un attaccante molto forte).

Una successiva applicazione della Gamification ad un altro aspetto del software testing si può trovare nello studio di (Santos et al., 2019), che presenta **Clean Game**, ossia un tool per ludicizzare la pratica di *code refactoring*.

Il *code refactoring* è una sotto-attività del software testing che ha lo scopo di individuare le fragilità del codice e sostituirle con un codice equivalente dal punto di vista funzionale, ma privo di tali imprecisioni: queste fragilità vengono chiamate in gergo *code smell*. I code smell sono principalmente artefatti noti assieme al proprio metodo di refactor, definiti in (Fowler, 1999). In Clean Game si sfrutta un analizzatore statico di codice sorgente chiamato **PMD**, che effettua il parsing del codice dato in input alla ricerca dei code smell noti.

Clean Game è diviso in due moduli: un quiz sui code smell a risposta multipla e un modulo di identificazione dei code smell.Gli elementi di Gamification usati in questi moduli sono: player status, punteggi, badge, tempo impiegato (per fornire le risposte nei quiz e nella ricerca dei code smell) e classifiche per ogni modulo aggiornate in tempo reale. Sono utilizzabili fino a tre suggerimenti per l'identificazione dei code smell e per i quiz, ma chiedere aiuto alla piattaforma penalizza il giocatore in termini di punteggio.

Clean Game può essere adottato non solo a livello accademico, ma anche a livello aziendale, in quanto i quiz vengono creati dinamicamente in modo automatico sulla base del codice fornito in input e, non essendoci la necessità di una figura intermedia che conosca il codice e i suoi code smell, l'integrazione con l'ambiente lavorativo può essere realizzata con costi contenuti.

Clean Game può inoltre essere completamente integrato con la GitHub API: alla creazione di una nuova stanza l'utente deve fornire l'URL della repository Java su GitHub su cui si vuole operare. Dopodiché, il codice è clonato e trasformato in un Abstract Sintax Tree (AST), usato come oracolo per le domande relative al codice.

L'esperimento condotto da (Santos et al., 2019) mostra come l'uso di tale sistema ludicizzato aiuti nell'identificazione dei code smell e nel refactor, portando ad una individuazione del doppio dei code smell rispetto alla loro ricerca in un ambiente non ludicizzato. Dalle valutazioni personali dei partecipanti sono emersi dati positivi di coinvolgimento eccetto che negli aspetti di focus (poca concentrazione potrebbe essere tuttavia dovuta all'eccessiva difficoltà dell'esperimento) e nella soddisfazione: i risultati sono stati migliori, ma è probabile che la Gamification non abbia avuto un design abbastanza coinvolgente da migliorare significativamente l'entusiasmo e il coinvolgimento dei partecipanti.

Va tuttavia sottolineato come il campione cui è stato somministrato l'esperimento fosse molto ristretto (18 persone) e limitato all'ambiente accademico. Non è dunque noto, ad oggi, se a livello aziendale l'impatto possa essere lo stesso.

(Costa & Oliveira, 2019), invece, descrivono un possibile tentativo di applicazione dei principi della Gamification all'insegnamento dell'exploratory testing, ossia un approccio di testing manuale che pone enfasi sulla libertà e la responsabilità del tester di esplorare il sistema.

L'attività di testing viene immersa in un'ambientazione piratesca, il cui tema principale è quello della caccia al tesoro: il codice viene diviso in aree da esplorare e i tester devono adottare delle strategie per scovare i tesori (ossia i bug).

Vengono utilizzati diversi elementi:

- *Profili*: identificano i diversi ruoli in gioco, come tester o esperti (ossia coloro che gestiscono il sistema di Gamification);

- *Avatar*: rappresenta lo studente nel gioco e può raggiungere diversi livelli a seconda del punteggio ottenuto; è inoltre presente un *general avatar* che rappresenta un team di studenti, ottenuto sostanzialmente facendo la media fra gli avatar dei componenti.

- *Attività*: sono i vari task che un tester può svolgere. Al completamento di ognuna di esse si ottiene un determinato *punteggio*. Ottenere un certo numero di punti durante le fasi del gioco può portare il tester ad ottenere delle *medaglie* (ossia dei badge) o delle *monete* (chiamate *bitskull*), che costituiscono il premio in caso di raggiungimento del punteggio massimo in una fase e permettono di comprare delle risorse (come le *carte*, descritte di seguito);
- *Carte*: sono oggetti equipaggiabili all'avatar di un tester. Sono disponibili tre tipi di carte:
  - *Difesa*: forniscono uno strumento per difendersi da eventuali attacchi nemici;
  - *Attacco*: permettono di attaccare i nemici (per esempio consultando i loro requisiti di sistema, parte del tutorial, ricevendo l'aiuto di uno specialista ecc..);
  - *Accessori*: permettono di personalizzare il proprio avatar.

All'inizio del corso, gli studenti (divisi in team) sono istruiti sulle dinamiche del sistema e viene loro raccontata una breve storia sull'ambientazione in modo da rendere il tutto più immersivo. Le attività svolte successivamente vengono invece suddivise nelle seguenti fasi (spiegate descrivendo uno scenario esemplificativo con due soli team):

1. Il team A e il team B hanno un certo lasso di tempo per cercare il maggior numero di bug possibile nel proprio codice.
2. Entrambi i team producono un report (*bug report*) su quanto fatto finora, descrivendo le strategie adottate ed eseguendo una prioritizzazione dei bug trovati.
3. Si svolge una fase di *battaglia*, dove i due team si scambiano e giudicano a vicenda i propri report prodotti nella Fase 2 e inseriscono i risultati dell'analisi in un altro report (*analysis report*).
4. Un esperto giudica i report prodotti dai due team, assegnando un punteggio in base al numero e alla qualità dei bug segnalati nel bug report e producendo un'analisi di tale resoconto, così da confrontarla con l'analysis report prodotto dall'altro team e assegnargli dunque un determinato punteggio. Prendendo

come esempio il solo team A, l'esperto assegnerà ad esso dei punti sulla base della qualità del bug report prodotto, per poi produrne un'analisi da confrontare con l'analysis report prodotto dal team B (e assegnare al team B un certo numero di punti).

Durante la consegna dei risultati, i team, in base al loro punteggio, possono ricevere una ricompensa ed eventualmente una carta (previa risoluzione di un puzzle aggiuntivo). I team che sono riusciti a portare il proprio general avatar all'ultimo livello, dopo la risoluzione di tre puzzle finali, possono essere ricompensati con un premio nascosto fisicamente all'interno della classe. Ciò enfatizza ancora di più l'atmosfera e il tema della caccia al tesoro, aggiungendo immersività.

Il risultato di questo approccio può essere valutato sulla base dei dati ricavati dalle varie attività, del feedback degli studenti e del confronto fra due form che devono essere sottoposti agli studenti prima e dopo il corso, per permettere di capire come siano cambiate le loro conoscenze grazie al corso stesso e all'utilizzo di un sistema gamificato.

1.2.5 Limiti

I casi di applicazione delle tecniche di Gamification hanno avuto una crescita significativa negli ultimi anni (Pedreira et al., 2015), tale da stimolarne un ampio utilizzo in ambito IT. In molti casi, però, la realizzazione e l'integrazione dei sistemi ludicizzati sono state eseguite in maniera inadeguata, evidenziando i limiti di applicabilità di questo approccio. Di seguito verranno elencati alcuni casi noti.

Nello studio realizzato da (Berkling & Thomas, 2013) e citato in precedenza, si evidenzia come dai feedback raccolti direttamente dagli studenti tramite dei questionari sia emersa una profonda disparità nelle aspettative riguardanti l'uso dei termini e degli elementi della Gamification in un ambito come quello universitario. Molti studenti (la maggior parte dei quali non ha definito sé stessa "gamer") ha individuato infatti nel semplice divertimento la principale ragione che li spinge a giocare solitamente, trascurando quasi completamente altri fattori, come ad esempio il senso di sfida, la sensazione di autonomia o la voglia di raggiungere degli obiettivi, che sono invece tra gli aspetti principali su cui fa leva la Gamification. Di conseguenza, la maggior parte degli studenti ha percepito tutti gli elementi ludici come inutili, noiosi o addirittura dannosi in certi casi.

Oltre a questo aspetto, sono state riscontrate anche delle problematiche legate all'eccessivo senso di autonomia generato negli studenti, che ha portato diversi di loro a pensare di non aver più bisogno dell'ausilio dei professori o addirittura di non avere necessità di seguire il corso stesso. Molti di loro, inoltre, hanno trascurato del tutto la maggior parte degli elementi maggiormente legati al gioco, dai più classici, come classifiche, punti, badge e livelli, ai meno comuni, come i punti extra ed il riconoscimento pubblico assegnati in caso di aiuto ad altri studenti.

Gli sviluppatori della piattaforma hanno individuato, tra le principali cause dietro al fallimento dell'esperimento, la mancanza di appeal della piattafroma dal punto di vista estetico, giungendo inoltre alla conclusione che la scelta migliore fosse quella di integrare gli elementi della Gamification nella piattaforma senza nominarli esplicitamente e di introdurre i cambiamenti in maniera più lenta e graduale.

Secondo lo studio di (Dal Sasso et al., 2017), uno dei grandi problemi dei sistemi ludicizzati è la *pointsification*, un problema già nominato in precedenza e che può presentarsi in diverse situazioni. Esso consiste nel tentativo da parte dell'utilizzatore del sistema ludicizzato di totalizzare un grande quantitativo di punti, trascurando il reale scopo dell'introduzione dei punteggi, ossia il completamento di un determinato lavoro.

Il giocatore finisce così per trascurare i compiti più difficili e remunerativi in termini di punteggio in favore di quelli più semplici. Questo comportamento è dovuto al fatto che i giocatori si focalizzano sulla sola attività di raccoglimento dei punti e, pur di avere una assegnazione di punteggio immediata, concentrano la loro attenzione sulle attività meno dispendiose a livello di impegno, risorse e tempo, per non dover attendere il completamento di un task molto complesso.

Un ulteriore aspetto spesso trascurato quando si tenta di fare una analisi costi/-benefici sull'uso della Gamification consiste nel fatto che non esiste alcuna ricerca sui suoi effetti a lungo termine; (Platonova & Bērziša, 2017) ipotizzano che a lungo andare si perda l'impatto positivo mostrato invece sul breve periodo, ma tale affermazione resta un'ipotesi che non può essere né confermata né smentita allo stato attuale delle cose.

Altre volte il sistema non porta vantaggi perché semplicemente mal implementato: è fondamentale che il layer di Gamification, oltre che essere appetibile dal punto di vista ludico e dell'intrattenimento, sia anche completamente privo di bug e glitch, per evitare che tali problemi compromettano l'esperienza dell'utente, rendendola meno gratificante e immersiva.

Nell'esperimento condotto da (Buckley & Clarke, 2018) si fa notare come una delle possibili ragioni del fallimento del sistema didattico ludicizzato proposto sia l'insorgenza di bug notata proprio dagli studenti appartenenti al gruppo sperimentale cui era sottoposta la piattaforma ludicizzata. Oltre a questa ragione viene menzionato anche il fatto che il gruppo di controllo potesse accedere al materiale fornito dal professore senza dover passare attraverso un meccanismo di identificazione, mentre al gruppo sperimentale era richiesta un'identificazione iniziale tramite login con username e password.

Altri sistemi ludicizzati hanno aspetti di applicabilità limitata: funzionano molto bene a livello accademico per l'insegnamento di una particolare attività, ma trovano difficoltà nell'applicazione in un ambiente aziendale. Tali difficoltà possono essere dovute alla complessità di integrazione con l'ambiente di lavoro (se un tool non è ben integrato porta alla distrazione dell'utilizzatore, che è forzato a fare un context switch costoso da un'attività all'altra perdendo concentrazione) (Prasetya et al., 2019) oppure al fatto che i tool, per essere applicati, necessitano di un overhead che, se in ambito accademico è normalmente demandato al professore, nell'ambiente lavorativo non ha un'attribuzione chiara e univoca: l'assegnazione del punteggio fatta automaticamente deve tenere conto della difficoltà del task (non banalmente quantificabile) e il completamento delle quest non sempre può essere determinato in modo automatico, ma necessita che sia un persona fisica a considerare una quest completa (Sheth et al., 2011).

1.3 GUI testing e tool esistenti

Il GUI testing è la disciplina che si occupa di testare l'applicazione utilizzando l'interfaccia grafica, interagendo con essa proprio come farebbe l'utente finale. Esistono principalmente due approcci al GUI testing: manuale e automatizzato.

Il metodo **manuale** richiede che sia il tester a effettuare i controlli eseguendo i test case a mano ad ogni iterazione. Ciò comporta un costo molto elevato soprattutto in termini di tempo speso nell'esecuzione dei test, unito ad una maggiore probabilità di commettere errori durante la sessione, essendo l'esecuzione fortemente legata a fattori umani.

L'approccio **automatizzato**, invece, si basa sulla creazione di script di test da parte del tester e sulla loro esecuzione automatica ad ogni iterazione. Gli script possono essere programmati direttamente in un linguaggio (o pseudolinguaggio)

di programmazione oppure creati in modalità *record & replay*, in cui il tester registra le operazioni definite nel test case una tantum, eseguendole manualmente. Successivamente, ad ogni iterazione, le interazioni registrate vengono emulate.

Questa modalità permette di risparmiare in termini di tempo e di risorse umane impiegate (Borjesson & Feldt, 2012), ma anche di ottenere prestazioni migliori in termini di coverage raggiunta e bug trovati. Il rovescio della medaglia è dato da una minor flessibilità e una forte fragilità dei test case automatici: infatti, ogni volta che la GUI subisce alterazioni, talvolta anche minime, i test potrebbero necessitare di un intervento umano per riprogrammarli. Inoltre, sebbene l'esecuzione automatizzata dei test sia più rapida se confrontata con l'esecuzione manuale, la creazione e la manutenzione dei test richiedono molto tempo.

Una nuova tipologia di testing automatizzato è l'**Augmented Testing** (AT), introdotto nello studio di (Nass et al., 2019) basandosi sulla tecnica già nota di *Record & Replay*. Questo nuovo metodo permette di svolgere l'attività di testing del *System Under Test* (SUT) usando la cosiddetta Augmented GUI: un'interfaccia che si sovrappone alla GUI del sistema, arricchendola con informazioni supplementari dedotte a partire dai metadati. La Augmented GUI fornisce suggerimenti al tester, evidenzia aree importanti dell'interfaccia e permette di registrare dei commenti.

Il principio su cui si basa questa tecnica è l'utilizzo di un *Augmented Layer* (AL) che si interpone tra il tester e il sistema testato, mediando le interazioni che avvengono fra queste due entità. L'AL riceve le informazioni sulla pagina da mostrare dal SUT ed elabora i metadati presenti, fondendo in una sola vista l'interfaccia originale con le informazioni aggiuntive dedotte (bottoni da cliccare, testo da controllare, caselle di testo da riempire ecc.).

Il tester interagisce con l'AL attraverso check, issue e azioni aumentate che captano l'interazione avvenuta e agiscono sul modello riproducendo a tutti gli effetti la stessa azione sull'interfaccia originale. Tutte le interazioni avvenute sono registrate durante la sessione ed evidenziate da un rettangolo colorato attorno al widget su cui è avvenuta l'interazione, consentendo al tester di selezionare un'azione precedentemente registrata per riprodurla oppure crearne una nuova.

L'Augmented Testing precedentemente descritto è implementato da **Scout**: un prototipo sviluppato in Java che applica tali nozioni per testare applicazioni web. Tale prototipo ha presenta alcune limitazioni nella presente implementazione, che verranno esposte nei capitoli successivi.

Negli studi di (Nass et al., 2019) e (Nass, Alégroth, & Feldt, 2020) viene mostrato come l'Augmented Testing abbia delle potenzialità elevate in quanto permette di creare test case in meno tempo e con più semplicità rispetto a tool ad oggi esistenti. Combinando gli effetti della Gamification a questa tecnica si mira a creare un ambiente di GUI testing efficiente, facile da imparare e piacevole per l'utilizzatore, al fine di ottenere un miglioramento delle prestazioni e un incremento della soddisfazione del tester.

Nelle sezioni successive si parlerà dello sviluppo di un plugin per Scout che implementi alcune tecniche di Gamification, delle quali verranno poi valutati gli effetti sortiti.

Il resto della tesi è strutturato come segue:

- il Capitolo 2 fornisce una descrizione dettagliata riguardo all'implementazione degli elementi di Gamification inseriti nel tool;
- il Capitolo 3 fornisce un esempio di sessione di testing svolta utilizzando Scout;
- il Capitolo 4 descrive la sperimentazione e il processo di validazione del plugin;
- il Capitolo 5 descrive i limiti di applicabilità dello studio;
- il Capitolo 6 presenta le riflessioni conclusive e suggerisce alcune possibilità di ampliamento del lavoro svolto.

Capitolo 2

Architettura e design

Per costruire un plugin che implementi la Gamification, è stato necessario analizzare il software già esistente nel dettaglio.

2.1 Scout

Scout (Nass et al., 2020) è un prototipo che implementa il concetto di Augmented Testing, sviluppato per valutare l'efficacia della tecnica in collaborazione con professionisti del settore industriale. Gli sviluppatori mantengono il core del tool confidenziale, dando però la possibilità di arricchire il software con nuove funzionalità tramite lo sviluppo di plugin. Alcuni di essi, nonostante siano di fatto opzionali, sono alla base del corretto funzionamento del software.

In Figura 2.1 viene mostrata una panoramica del tool, comprensiva dei moduli già presenti nel codice originale (non colorati all'interno del grafico), di quelli già presenti ma modificati o adattati per permettere l'implementazione degli elementi ludici (evidenziati in arancione) e, infine, dei moduli aggiunti ex novo (evidenziati in verde).

Come si può vedere, Scout memorizza al proprio interno un *Application State* che modella lo stato del SUT con tutte le dipendenze interne necessarie al funzionamento e le varie interazioni avvenute. A partire dall'home state, viene creata una struttura che rappresenta lo *State Model* (Fig. 2.2), contenente gli stati visitati, che può essere attraversato e modificato man mano nelle operazioni di testing.

L'*Augmented GUI* (AGUI) è l'unica interfaccia grafica mostrata al tester, mentre l'effettiva GUI del SUT non è direttamente interagibile. Ciò implica che tutte

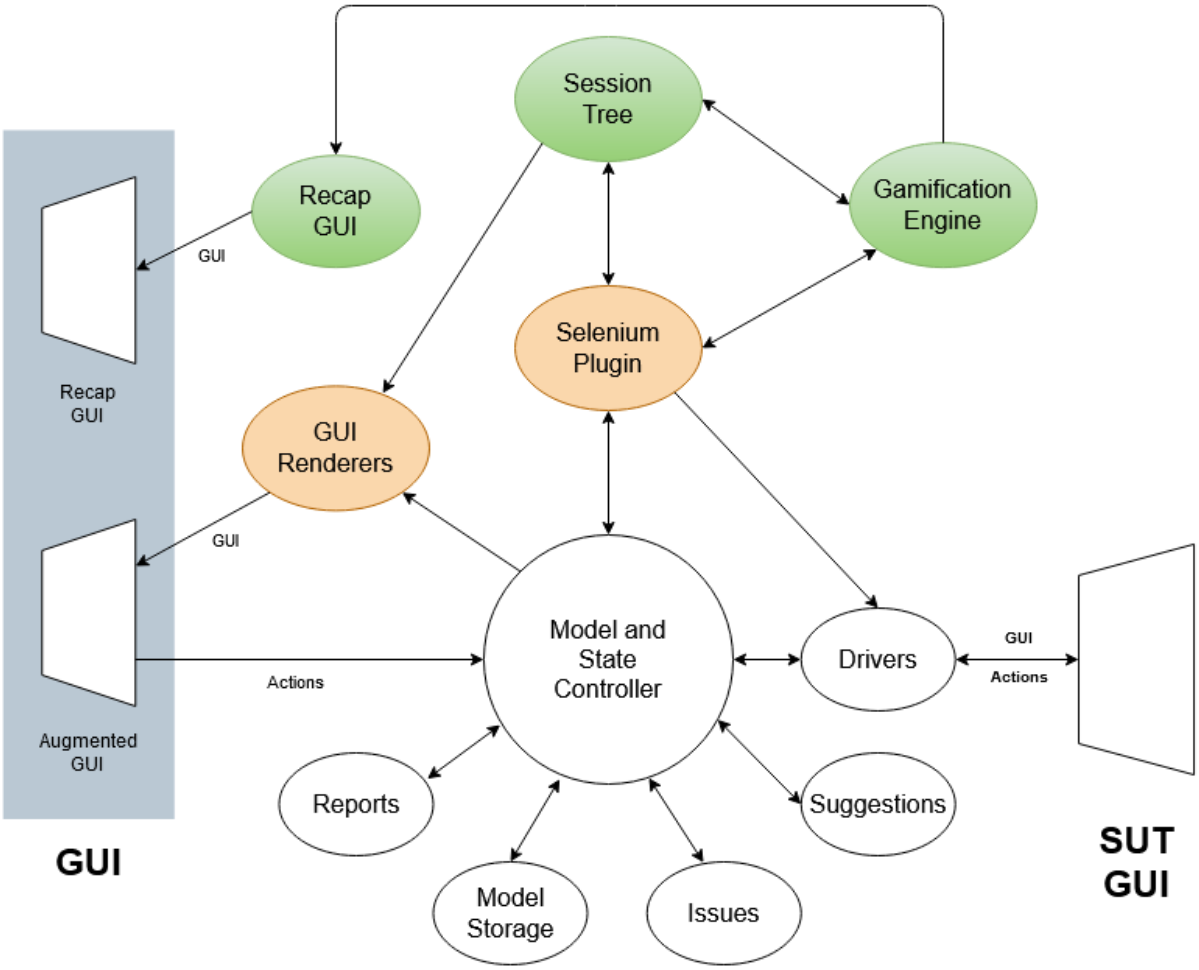

Figura 2.1. Panoramica del tool.

le azioni eseguite dal tester siano ricevute dall'AGUI, mappate come interazioni aumentate e trasferite alla GUI del SUT tramite un *driver* che le traduce nelle corrispondenti azioni da effettuare. L'AGUI arricchisce la GUI del SUT con elementi aggiuntivi come suggerimenti e azioni aumentate (check, issue, click, ecc.), che vengono renderizzati tramite le classi Java *AugmentState* e *AugmentToolbar*.

La prima riceve le informazioni sullo stato del SUT dallo State Model e traduce i metadati in oggetti visivi, i quali contornano le zone della GUI in cui sono avvenute le interazioni con un colore dipendente dal tipo dell'interazione stessa: i check sono contornati in verde, i click in blu, gli issue in rosso e le espressioni che devono ancora essere valutate in giallo.

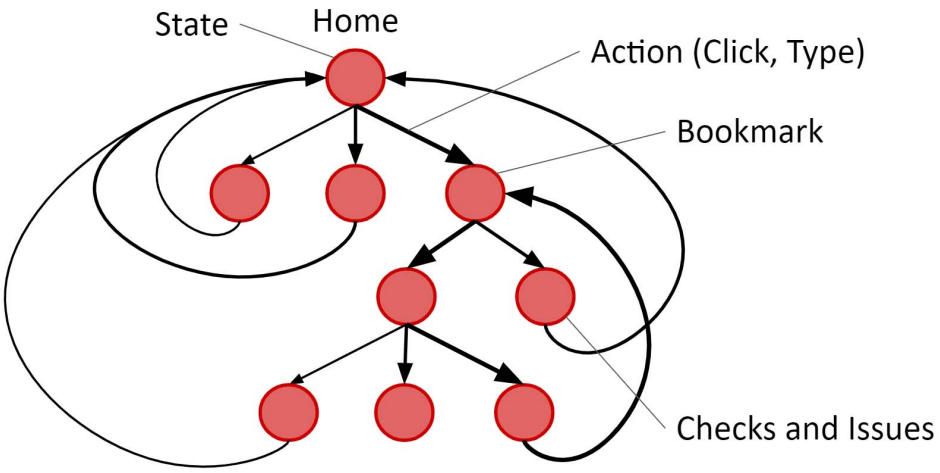

Figura 2.2. State Model.

La seconda, invece, permette al tester di configurare Scout (scegliendo i plugin da attivare) e di gestire la sessione corrente di testing, avviandola, interrompendola o salvandone lo stato. Inoltre, AugmentToolbar permette la navigazione nell'albero degli stati del SUT, consentendo di tornare a quello iniziale o ad uno precedentemente salvato in un bookmark.

Tutte le azioni catturate dall'AGUI sono salvate nello State Model, implementato come grafo pesato, in cui ogni nodo rappresenta un particolare stato dell'applicazione. I nodi sono tra loro collegati tramite le azioni eseguite dal tester, le quali, modificando lo stato, determinano un avanzamento nel grafo. I vari stati sono memorizzati all'interno del componente State Controller, che tiene traccia dell'intera struttura. Mantenere la sincronizzazione tra lo stato del SUT e il nodo corrente del grafo è particolarmente importante, poiché tutte le informazioni riguardanti l'applicazione sono memorizzate proprio in State Controller.

I driver, infine, sono essenziali in quanto elementi di congiunzione tra l'applicazione web, che deve essere necessariamente interpretata da un browser, e Scout. Sono proprio i driver, infatti, che permettono di catturare la pagina web da mostrare nell'AL.

2.2 Descrizione dei file di log

Come punto di partenza dello sviluppo del plugin, è stato creato un sistema di logging che annotasse ogni interazione, descrivendola nel contesto della pagina in cui si verifica ed arricchendola con alcuni metadati temporali e altri legati ai widget. Il modello descritto nella Sezione 2.3 è stato sviluppato in seguito all'analisi dei file di log prodotti nelle sessioni di esempio, svolte utilizzando il tool "as is", di cui sono state esaminate le interazini raccolte, al fine di produrre manualmente le quantità grezze descritte in seguito.

Vengono di seguito descritti i dati prodotti dal logger alla fine di ogni sessione di GUI testing.

Per ogni pagina web ne vengono mostrati l'URL, il tempo totale in millisecondi che il tester ha trascorso in tale pagina e infine l'elenco delle interazioni avvenute, rappresentate tramite delle stringhe formattate nel seguente modo:

<tipoInterazione> <idInterazione> <timestamp>

tipoInterazione specifica la tipologia dell'interazione registrata, che può essere una tra le seguenti:

- *Check*: indica che il tester ha richiesto un controllo sul contenuto del widget.
- *Issue*: indica che il tester si aspettava qualcosa di diverso dal widget selezionato.
- *Click*: registra un click su un widget. Solitamente questo porta al caricamento di un'altra pagina web.
- *Type*: annota la scrittura di una stringa all'interno di una casella di testo. Oltre alla tipologia di interazione, il logger annota anche la stringa digitata dall'utente.
- *Go Home*: registra se l'utente abbia utilizzato la funzione di Scout per tornare alla home del dominio da testare.
- *Select*: annota la selezione di una tra le opzioni di una drop-down list. Ogni volta che si cambia l'elemento selezionato, viene prodotta un'interazione di questo tipo.

Dopo aver specificato la tipologia dell'interazione registrata, viene indicato **idInterazione**, ossia un identificativo del web element sul quale è avvenuta l'interazione. L'assegnazione di una stringa univoca all'interno della pagina si basa sui metadati HTML assegnati all'elemento.

Le tipologie di elementi che compongono una pagina HTML sono innumerevoli e dipendono dal dominio su cui si effettua GUI testing. Allo stesso modo, anche la presenza e la descrittività dei campi che compongono un tag HTML sono peculiari per ogni sito web e dipendono dalle scelte effettuate dal programmatore che lo ha sviluppato.

Per queste ragioni, si è scelto di imporre una priorità nell'assegnazione dell'identificativo, seguendo quest'ordine di attribuzione:

1. Campo ID (se disponibile).
2. Nome del widget (se assegnato).
3. Classname (se l'elemento appartiene ad una classe).
4. Tipo del widget (se assegnato).
5. Riferimento ipertestuale (se presente).
6. Hash code della stringa ottenuta concatenando il campo value, il campo text e il tag dell'elemento (se nessuna delle condizioni precedenti è soddisfatta).

Infine, l'ultima componente del log di un'interazione è **timestamp**, ossia l'istante di tempo alla quale essa avviene, espresso in millisecondi.

2.3 Rappresentazione della sessione

All'avvio del software, è possibile iniziare una sessione di testing con l'apposito pulsante, indicando un ID che rappresenti il tester che vuole effettuare la sessione, la pagina web di partenza (*Home Locator*) e il browser che si vuole emulare.

La rappresentazione di una sessione di testing è resa possibile dall'introduzione di quattro classi fortemente interconnesse, raggruppate nel package rappresentato in Figura 2.3. La singola sessione è stata rappresentata tramite un oggetto di

classe *Session*, che modella una struttura dati ad albero non binario. Ogni nodo dell'albero è un oggetto di classe *Node*, ossia un wrapper di una pagina web, unita ad altri attributi che consentono la navigazione dell'albero stesso, quali il nodo padre (attributo *father*) e una lista dei nodi figli (attributo *children*).

La classe **Session** permette di gestire ad alto livello la struttura, mantenendo un riferimento *root* alla radice dell'albero (così da permettere di registrare correttamente l'operazione di ritorno all'home locator, che Scout consente di effettuare tramite il comando *Go Home*) ed un altro, chiamato *current*, al nodo corrente in cui si trova il tester ad un certo punto della propria sessione di testing. *Session* fornisce inoltre alcuni metodi per l'aggiunta di nodi all'albero (sempre a partire dal nodo indicato dall'attributo *current*), uno per visitare l'albero ricorsivamente ed infine un metodo che ritorna un array list contenente tutte le coverage raggiunte nelle pagine visitate nel corso della sessione, utile per calcolare delle metriche che verranno spiegate nelle sezioni successive.

Come detto in precedenza, ogni oggetto **Node** contiene al suo interno una pagina web, modellata tramite la classe **Page**, le cui istanze contengono un riferimento allo stato di Scout nell'attributo *scoutState* (istanza della classe *AppState*), necessario per il ripristino dello stato tramite lo State Controller quando si ritorna su una pagina già visitata, la lista delle interazioni che si sono verificate all'interno della pagina (*interactions*), il numero di widget interagibili (*totalWidgets*), il numero di widget coi quali il tester ha interagito (*highlightedWidgets*) ed un meccanismo di annotazione del tempo trascorso all'interno della pagina, basato sull'utilizzo della classe *Timing*.

Timing modella un timer che definisce un solo intervallo temporale con un determinato inizio ed una determinata fine, senza fornire la possibilità di farlo ripartire una volta fermato. La classe mette a disposizione un metodo *computeTime*() che calcola la differenza fra l'istante finale e l'istante iniziale e ritorna il numero di millisecondi trascorsi fra i due. Dato che una stessa pagina può essere visitata più volte nel corso di una stessa sessione, ogni oggetto *Page* contiene un array list di oggetti *Timing*, ognuno dei quali rappresenta la durata di una certa visita.

Ogni sessione viene identificata dall'ID del tester e dall'istante temporale in cui si conclude. Questo meccanismo fa sì che tutte le sessioni possano essere salvate e utilizzate come base per calcolare statistiche puntuali o aggregate. In particolare, al termine della sessione, le apposite classi *StatsComputer* (spiegata nel dettaglio nella Sezione 2.6) e *Timing* permettono di calcolare una serie di dati grezzi che

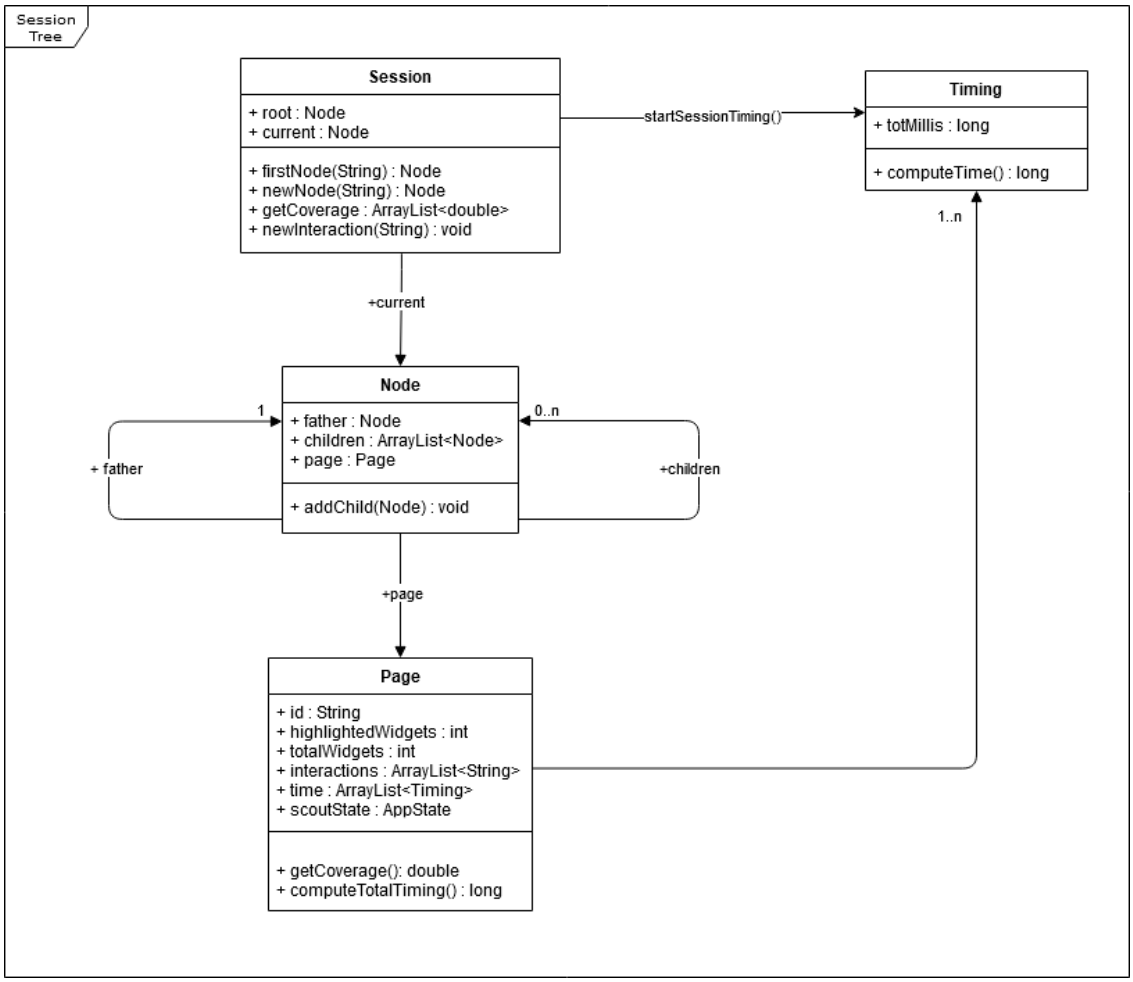

Figura 2.3. Package contenente le classi che permettono di rappresentare una sessione all'interno del plugin.

forniscono delle informazioni riguardo alla sessione appena conclusa. Tra i dati più rilevanti troviamo:

- Numero totale di widget cliccabili individuati nelle pagine che compongono la sessione appena conclusa. Questo dato si ottiene visitando ricorsivamente l'albero che modella la sessione, sommando l'attributo *totalWidgets* di ogni pagina presente al suo interno.

- Numero totale di widget con cui il tester ha interagito durante la sessione. Anche questo dato viene calcolato visitando ricorsivamente l'albero e sommando l'attributo *highlightedWidgets* di ogni pagina visitata nella sessione.

- Numero di interazioni avvenute per ogni pagina. Tale dato si ottiene contando tutte le interazioni avvenute sui widget della pagina. A differenza del numero totale dei widget con cui si ha interagito, in questo caso si possono verificare più interazioni su uno stesso widget in momenti diversi della sessione (repliche identiche dell'azione o interazioni con parametri diversi).

- Numero di pagine visitate, definito come la cardinalità dell'insieme delle pagine identificate dal proprio URI.

- Tempo trascorso all'interno di una singola pagina, espresso in millisecondi. Si noti che, avendo la possibilità di ritornare alla radice dell'albero (tramite l'apposito tasto *Go Home* messo a disposizione da Scout), alcune pagine possono essere visitate più volte all'interno della stessa sessione. Per questo motivo, ogni oggetto *Page* contiene una lista di oggetti *Timing* i cui valori, sommati, restituiscono il tempo totale di permanenza all'interno di una stessa pagina nell'arco dell'intera sessione.

- Tempo totale impiegato dal tester per portare a termine la sessione. Viene calcolato inizializzando un oggetto *Timing* all'inizio della sessione (pressione del tasto *Start*) e fermando il conteggio alla sua conclusione (pressione del tasto *Stop*).

2.4 Estrapolazione delle metriche

Utilizzando e combinando i dati grezzi precedentemente riportati, si può estrapolare una serie di metriche che descrivono la sessione. Tali metriche sono alla base dell'utilizzo degli elementi di Gamification introdotti nel plugin.

Le metriche prodotte per ogni sessione sono le seguenti:

- La **Coverage raggiunta su una singola pagina**, ottenuta come rapporto fra il numero di *highlighted widget* e quello di *total widget* della pagina.

- La **Coverage totale della sessione**, ottenuta calcolando la media delle coverage raggiunte nelle singole pagine che compongono la sessione. Se una pagina visitata non ha subìto nessuna interazione, questa viene esclusa dal computo della coverage per evitare di alterare la reale coverage della sessione con pagine delle quali non si ha interesse.

- Il **Tempo medio per interazione**, ottenuto calcolando il rapporto tra la durata in secondi della sessione e il totale delle interazioni avvenute in ogni pagina.

- Il **Numero medio di interazioni per pagina**, ottenuto calcolando il rapporto tra il totale delle interazioni avvenute su tutte le pagine e il numero di pagine visitate.

- Il **Numero di pagine scoperte per la prima volta**, ossia il numero di pagine che non erano mai state visitate fino a quel momento in una determinata sessione. Per calcolare questo numero viene conservato l'insieme delle pagine visitate in modo permanente nello stato dell'applicazione e, al termine della sessione, viene fatta la differenza insiemistica tra le pagine visitate nella presente sessione e quelle già visitate in precedenza.

- Il **Numero di widget che sono stati scoperti per la prima volta**. In questo caso vengono memorizzati in modo permanente tutti i widget su cui sia avvenuta un'interazione per ogni pagina. Man mano che il tester agisce sui widget durante la sessione, si controlla se in tale pagina si fosse già verificata un'interazione con quel particolare widget.

- Il **Numero di issue** riportati durante la sessione. Gli issue sono i problemi verificatisi nel sistema, riconosciuti e segnalati dal tester, tipicamente con un commento.

- Il **Numero di bug noti**, ovvero il totale di errori causati da fattori esterni. Questi bug descrivono errori generici o temporanei che si possono verificare nella rete. Questo tipo di problematiche è conosciuto a priori dal tester assieme al corrispondente messaggio d'errore, che va esplicitato in un file di configurazione. Quanto scritto all'interno di questo file è necessario al tool per il riconoscimento dei messaggi d'errore, che compaiono a schermo nel caso in cui si verifichino problemi di vario genere (errori di connessione, impossibilità di localizzare una risorsa, ecc.).

- Il **Numero di easter egg** costituisce una metrica che, seppur piuttosto inusuale, indica con quanta profondità sia stato esplorato il dominio. Tale valore è un numero assoluto inversamente proporzionale al numero di collegamenti che sussistono tra le pagine. Considerando il SUT come un grafo orientato

non pesato, le varie pagine che costituiscono l'applicazione sono i nodi del grafo e i collegamenti tra le varie pagine sono gli archi. Per ogni nodo esiste un solo arco che, se percorso, genera l'easter egg. Aumentando i collegamenti tra le pagine (gli archi del grafo), diminuisce la probabilità di percorrere esattamente l'arco che porti alla generazione dell'easter egg.

2.5 Adattamento del tool

Come si evince dalla Figura 2.1, il principale punto di collegamento tra Scout e i moduli che implementano la Gamification è la classe Java **SeleniumPlugin**.

Questa classe, che da principio aveva il ruolo di legante tra la logica degli stati di Scout e le invocazioni delle funzioni dei webdriver, è stata arricchita con la logica di Gamification e di raccolta dei dati.

All'avvio della sessione di *SeleniumPlugin*, oltre all'inizializzazione del driver e alla creazione del Model interno, viene eseguita anche l'inizializzazione della sessione di Gamification con la creazione di un nuovo albero delle pagine, istanziando un oggetto di classe *Session*.

SeleniumPlugin possiede il metodo *verifyAndReplace()*, che permette di gestire i cicli di refresh della schermata: in questo metodo vengono controllati i widget in evidenza (contenuti nello State Controller) che devono essere mostrati nell'AL. In questa fase, vengono eseguito un filtraggio in modo da considerare soltanto i widget che rientrano nel campo di visibilità della schermata di Scout e quelli che hanno una dimensione non nulla, ma effettivamente cliccabile. Quando questo metodo è chiamato per la prima volta su una pagina nuova, si sceglie in modo casuale uno tra i link cliccabili presenti, il quale, se cliccato, conduce ad una pagina in cui è presente un *easter egg* (un elemento ludico spiegato in seguito nella Sezione 2.7.4).

Per ogni interazione verificatasi e captata da *SeleniumPlugin*, grazie al Gamification Engine viene calcolata una stringa descrittiva del tipo di azione accaduta, la quale, invocando il metodo *newInteraction()*, viene salvata all'interno della rappresentazione ludicizzata della pagina, controllando se l'interazione con il widget corrispondente sia già avvenuta in passato oppure sia avvenuta per la prima volta.

La maggior parte delle modifiche apportate a *SeleniumPlugin* riguarda le operazioni che si verificano ad ogni click su un link presente nella pagina. Se il link è valido, si procede con le operazioni che coinvolgono le classi dei package Gamification Engine e Session Tree: per prima cosa viene interrotto il timer relativo alla

pagina corrente; dopodiché, vengono salvati nel database i dati relativi ai widget coi quali nessuno aveva interagito finora e viene salvata la miglior coverage assoluta della pagina come highscore.

Utilizzando l'hyperlink come chiave, si verifica se la pagina web corrispondente sia già presente nell'albero di sessione mediante una visita ricorsiva a partire dalla radice. In caso contrario, si procede alla creazione di un nuovo nodo, figlio di quello corrente, contentente le informazioni riguardanti la nuova pagina. Nel primo caso viene ripristinato lo stato di Scout temporalmente più recente associato alla pagina. Viceversa, se la pagina viene inserita per la prima volta nell'albero di sessione, viene creato un nuovo stato ad essa associato.

In fase terminazione, prima di riportare il click del link, si verifica se la nuova pagina sia quella estratta per la generazione di un easter egg e, in tal caso, si procede all'assegnazione di coordinate casuali per il posizionamento a schermo dell'elemento grafico relativo all'easter egg. Dopo aver finalmente riprodotto il click del link tramite il webdriver, viene fatto ripartire il timer della pagina appena aperta.

Infine, dopo la chiusura del driver, vengono arrestati i timer attivi, calcolati tutti i dati necessari per la schermata di riepilogo della sessione ludicizzata (mostrata grazie alla classe *RecapGUI*) e, infine, salvati in modo permanente nel database i dati che riguardano quest'ultima (come highscore delle pagine, widget in cui si ha interagito, file di log dell'utente ecc.).

2.6 Gamification Engine

Le principali funzionalità relative agli elementi di Gamification sono state implementate all'interno del package **Gamification Engine**, rappresentato nel dettaglio in Figura 2.4. Tutte le classi presenti al suo interno hanno il compito di elaborare le metriche relative alla sessione appena conclusa e di renderle persistenti, eventualmente caricando prima in memoria quelle già presenti all'interno del database, qualora il tester avesse già eseguito delle sessioni di testing precedentemente, per aggiornarle alla luce dei nuovi risultati conseguiti.

Queste funzionalità vengono implementate con l'ausilio di due classi fondamentali e fortemente interconnesse: **StatsComputer** e **Stats**.

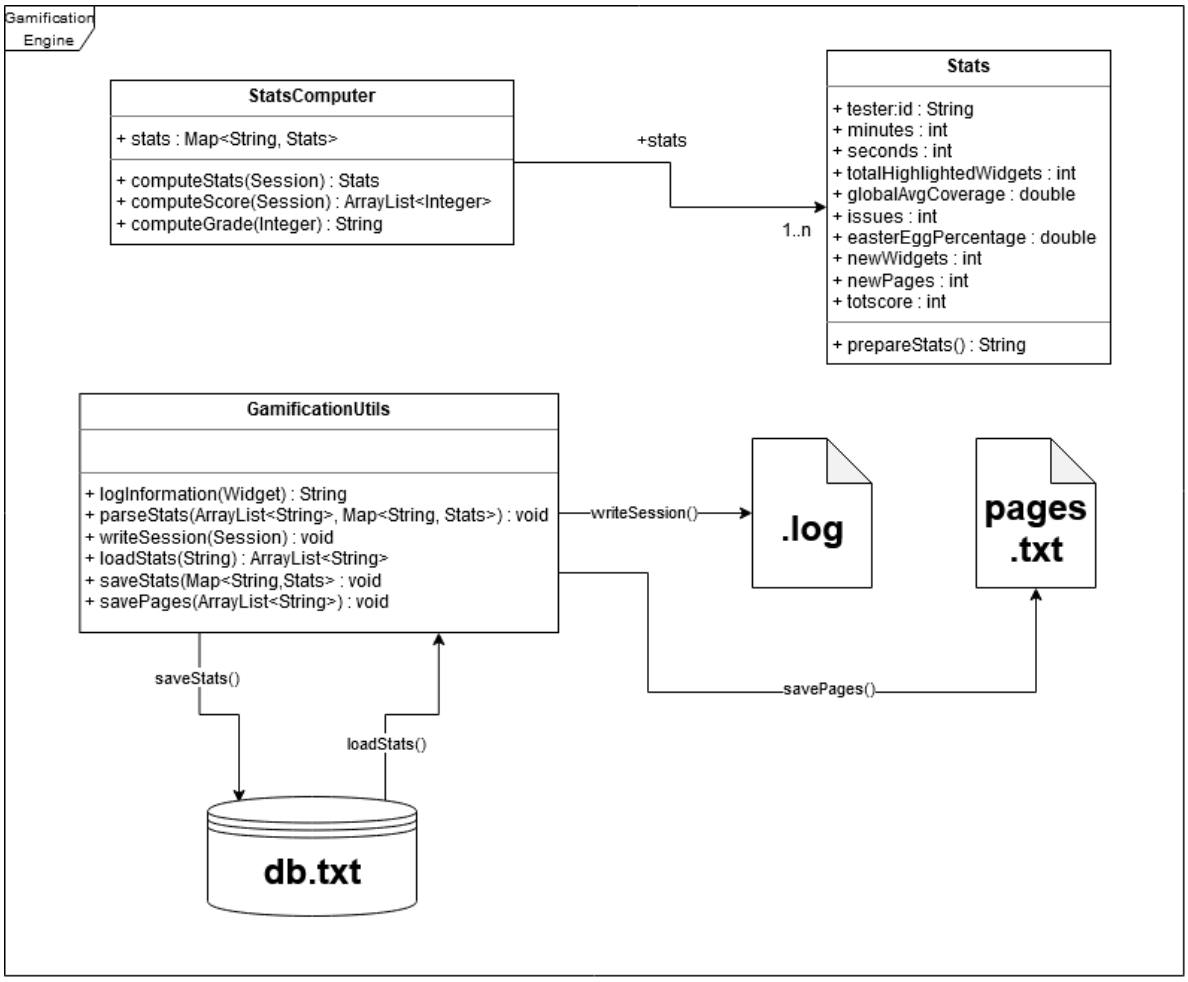

Figura 2.4. Package contenente le classi che implementano le funzionalità relative agli elementi di Gamification.

Ad ogni tester è associata una serie di statistiche che esso può costantemente aggiornare eseguendo più sessioni di testing all'interno della piattaforma. Tutte queste statistiche vengono salvate come attributi di un oggetto *Stats* associato a quel tester, ognuna delle quali permette di capire l'andamento dell'utente nell'uso del tool. Tali statistiche (legate alle metriche spiegate nella sezione precedente) sono:

- ID del tester;
- minuti e secondi totali trascorsi svolgendo delle sessioni di testing;

- numero totale di widget con cui il tester ha interagito nel corso delle varie sessioni;
- coverage media raggiunta nel corso delle varie sessioni;
- numero totale di issue segnalati;
- percentuale di easter egg trovati;
- numero totale di widget e pagine scoperti dal tester in esame;
- punteggio totale ottenuto nel corso delle varie sessioni svolte.

All'inizio di ogni sessione, viene istanziata la classe singleton *StastComputer*, che mantiene al suo interno una mappa che rispecchia il contenuto del database, associando ad ogni tester ID il corrispettivo oggetto *Stats*. Tale mappa viene creata al momento dell'istanziazione dell'oggetto di classe *StatsComputer*, andando a recuperare dal database (file *db.txt*) le informazioni necessarie, con l'ausilio della classe *GamificationUtils* (spiegata nel dettaglio successivamente). All'interno del database, ogni utente possiede una propria entry corrispondente ai dati del proprio oggetto *Stats*. Si noti che la classe *Stats*, inoltre, fornisce un metodo che permette di organizzare le informazioni contenute negli attributi dell'oggetto in un formato conforme alla sintassi utilizzata all'interno del database. In questo modo, sarà sempre possibile parsificare il file *db.txt* correttamente ad ogni istanziazione di *StatsComputer*.

La principali funzionalità della classe *StatsComputer* consistono nel calcolo delle metriche relative alla sessione corrente (metodo *computeStats()*), nella loro aggregazione con quelle già presenti nella mappa delle statistiche e nel calcolo del punteggio finale e del voto corrispondente (metodi *computeScore()* e *computeGrade()*). In particolare:

- *computeStats()*: metodo che permette di estrapolare le metriche a partire dai dati grezzi relativi all'oggetto *Session* che viene passato come parametro. Questo metodo calcola tutte le statistiche necessarie e controlla se il tester corrente possieda già un'entry all'interno del database (ossia se sia presente nella mappa *stats*): in caso affermativo, aggiorna l'entry corrispondente all'interno della mappa, altrimenti ne crea una nuova e la aggiunge alla struttura dati.

- *computeScore()*: metodo che si occupa del calcolo delle varie componenti del punteggio finale (spiegato nel dettaglio nella Sezione 2.7.2), sulla base delle metriche relative alla sessione. Il valore di ritorno consiste in una coppia di interi che rappresentano, rispettivamente, il punteggio base e il punteggio bonus totalizzati dal tester.

- *computeGrade()*: metodo che determina il voto da assegnare alla performance del tester, sulla base del punteggio complessivo ottenuto.

Il modulo che impelementa il Gamification Engine contiene inoltre una classe di utility chiamata *GamificationUtils*, che fornisce una serie di metodi statici che consentono di eseguire diverse operazioni di lettura e scrittura di dati sui file che permettono di rendere persistenti le informazioni relative alle varie sessioni.

Le principali funzionalità messe a disposizione dalla classe sono:

- Creazione di una stringa che descriva un'interazione appena avvenuta, in base al tipo di widget con cui il tester ha interagito. Ciò è reso possibile dal metodo *logInformation()*. Tutte le interazioni così codificate vengono inserite all'interno di un array list dell'oggetto *Page* corrispondente, una volta terminata la sessione (quindi all'interno del metodo *stopSession()* di *SeleniumPlugin*), con una chiamata del metodo *writeSession()*. Tale metodo va a creare un file di log (il cui nome viene generato a partire dall'ID del tester e dal timestamp relativo alla conclusione della sessione) che viene salvato all'interno di una cartella avente come nome l'ID del tester, posizionata nella directory */Gamification/*.

- Caricamento di dati e statistiche. Quest'operazione è resa possibile dal metodo *loadStats()*, che permette di leggere le informazioni contenute in un file il cui path viene passato al metodo come argomento. Tale metodo viene invocato all'avvio di una nuova sessione (all'interno del metodo *startSession()* di *SeleniumPlugin*) per caricare in memoria l'elenco dei messaggi di errore conosciuti a priori (contenuti nel file *BUGCONF_FILE*, interamente configurabile dal tester). *loadStats()* viene inoltre invocato nell'ambito dell'istanziazione della classe singleton *StatsComputer*, dove viene utilizzato per caricare in memoria il database contenente le statistiche di tutti i tester che hanno già compiuto almento una sessione di testing fino a questo momento

(file *db.txt*). Tali informazioni, una volta caricate in memoria, vengono parsificate tramite il metodo *parseStats()*, che permette di creare un oggetto *Stats* per ogni utente presente nel database e di associarlo all'ID del tester a cui fa riferimento, rendendo possibile la creazione di una mappa che viene salvata nell'attributo corrispondente all'interno dell'istanza corrente di *StatsComputer*. Infine, *loadStats()* viene utilizzato anche per caricare in memoria, una tantum, l'elenco delle pagine già visitate in precedenza da almeno un tester che abbia completato una o più sessioni prima di quella corrente. Tale operazione viene eseguita invocando il metodo all'interno del costruttore dell'oggetto *Session* relativo alla sessione corrente, passando come parametro il path del file *pages.txt*.

- Salvataggio delle statistiche in memoria. Quest'operazione viene eseguita al termine della sessione grazie al metodo *saveStats*(), una volta che *StatsComputer* ha terminato di aggiornare la mappa delle statistiche alla luce dei risultati ottenuti dal tester che ha completato la sessione corrente.

- Salvataggio dell'elenco contenente le pagine visitate da tutti i tester. Quest'operazione viene eseguita, dopo una serie di chiamate, all'interno di un metodo di *StatsComputer*, utilizzando il metodo statico *savePages()* di *GamificationUtils* e permette di rendere persistente l'elenco aggiornato delle pagine esplorate da tutti i tester che hanno eseguito almeno una sessione di testing (inclusa quella corrente, appena terminata).

2.7 Elementi di Gamification applicati

Durante la selezione ed implementazione delle metriche, sono state effettuate numerose sessioni di prova su alcuni domini ben noti. Le metriche ottenute durante queste sessioni hanno suggerito l'implementazione di alcuni elementi di Gamification, alcuni dei quali sono stati applicati fin da subito, altri solo teorizzati (riportati nella Sezione 2.7.6), lasciati ad eventuali sviluppi futuri.

2.7.1 Barra di progresso

Il primo elemento applicato è stata una barra di progresso che mostrasse visivamente il grado di completezza, in termini di coverage, raggiunto dal tester nella pagina

corrente.

In ogni pagina viene mostrata nella parte superiore della schermata una sottile linea rossa che si adatta in modo da occupare tutta l'ampiezza della schermata. Tale linea rappresenta la quantità di widget rimanente in percentuale per ottenere una page coverage del 100%. Una linea verde sovrasta quella rossa per rappresentare la coverage percentuale raggiunta in un determinato momento nella pagina come si evince in Figura 2.5.

Ad ogni interazione registrata su un nuovo widget, la barra verde di avanzamento viene aggiornata in tempo reale: ciò permette al tester di capire quanto sia profonda la sua esplorazione della pagina in esame e di intuire spannometricamente quanti widget interagibili siano presenti nel nodo corrente.

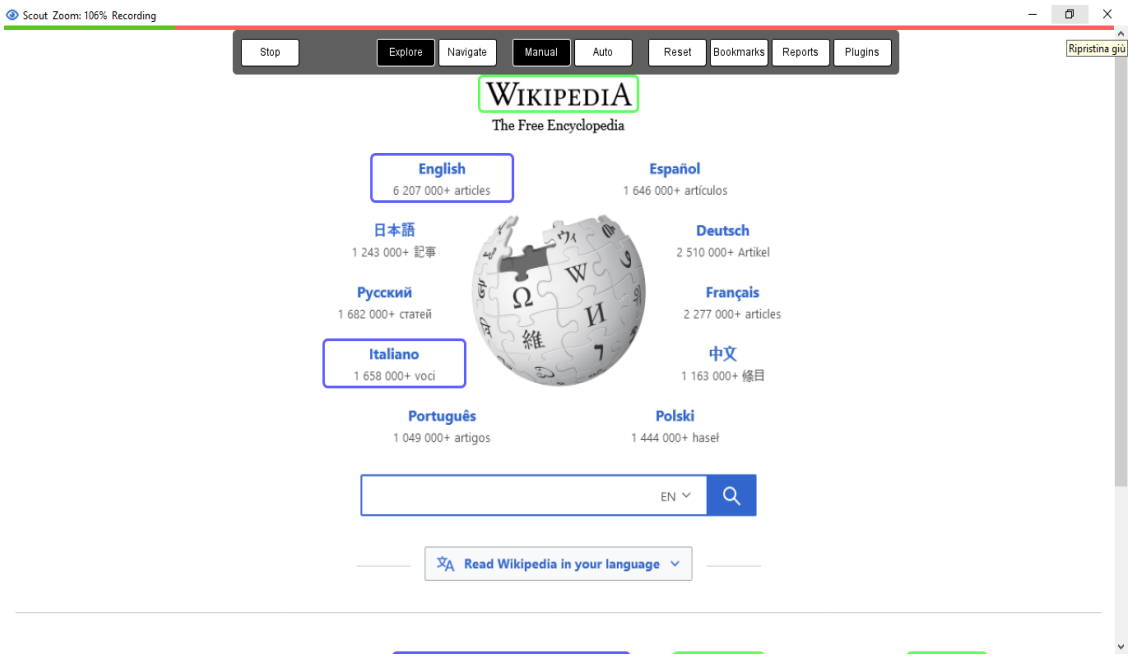

Figura 2.5. Barra di progresso di una sessione di testing nella pagina di Wikipedia.

Se almeno un altro tester ha già visitato la pagina corrente prima del tester corrente, la barra di progresso mostra anche la miglior coverage mai raggiunta nella pagina, visualizzata con una linea blu che si interpone tra quella verde e quella rossa: ciò indica il record di page coverage raggiunto sulla pagina e cerca di incentivare il tester a fornire una performance migliore rispetto a chi l'ha preceduto. Se il tester corrente supera il record, la linea verde sovrasta in estensione quella blu.

Le informazioni che si traggono da tale elemento ludico sono solamente quantitative, poiché esso non fornisce nessuna informazione riguardante collocazione e tipologia dei widget. Gli obiettivi di tale elemento sono quelli di rendere consapevole il tester dei progressi ottenuti sulla pagina corrente, di trasmettere un senso di soddisfazione al raggiungimento della coverage massima e, infine, di suscitare nel tester un lieve senso di sfida nei confronti dei tester passati. La barra di progresso nella sua interezza è mostrata in Figura 2.6.

Figura 2.6. Barra di progresso di una sessione di testing nella pagina di Wikipedia.

2.7.2 Punteggio

Al termine di una sessione, al tester viene mostrata una schermata che presenta i risultati ottenuti durante la sessione stessa. Vengono dunque mostrate le metriche principali (descritte nella Sezione 2.4), oltre ad un **punteggio** che riassume la performance del tester.

Il punteggio viene calcolato prendendo in considerazione diverse metriche e combinandole in modo da ottenere 3 componenti principali e 2 componenti bonus,

secondo la seguente formula:

$$\mathbf{P = a \cdot C + b \cdot EX + c \cdot EF + [d \cdot T] + [e \cdot PR]}$$

Le prime tre componenti costituiscono il **punteggio base** e ognuna di esse è pesata con il proprio coefficiente. In particolare:

- **C**: rappresenta la componente di **coverage** e descrive il grado di copertura medio raggiunto dal tester nelle varie pagine visitate durante la sessione, calcolata in termini di widget interagiti. In particolare, C viene calcolata come:

  $$C = \frac{\sum_{\forall i \in P} pc_i}{|P|}$$

  dove al numeratore troviamo la sommatoria delle coverage (pc_i) raggiunte nelle singole pagine dell'insieme P e al denominatore la cardinalità dell'insieme stesso. La coverage relativa ad una singola pagina viene a sua volta calcolata con il seguente rapporto:

  $$pc_i = \frac{w_{hl,i}}{w_{tot,i}}$$

  dove $w_{hl,i}$ è il numero di widget diversi con cui il tester ha interagito all'interno dell'i-esima pagina del set P, mentre $w_{tot,i}$ è il numero totale di widget cliccabili presenti nella medesima pagina i.

- **EX**: rappresenta la componente **esplorativa** della sessione e si basa sul numero di pagine che il tester considerato ha visitato e che non erano ancora state visitate da nessun altro fino a quel momento. La componente, inoltre, prende anche in considerazione i widget con cui il tester ha interagito per primo, anche all'interno di pagine già visitate in precedenza da almeno un collega. EX_{comp}, nel dettaglio, viene calcolata come:

  $$EX = \frac{k}{b} \cdot \frac{p_{new}}{p_{tot}} + \frac{h}{b} \cdot \frac{hw_{new}}{hw_{tot}}, \qquad k + h = b;$$

  dove il primo rapporto rappresenta la sottocomponente relativa alle nuove pagine scoperte (percentuale delle pagine nuove rispetto al totale delle pagine visitate durante la sessione), mentre il secondo quella relativa ai widget (percentuale degli *highlighted widget* nuovi rispetto al totale dei widget con

cui il tester ha interagito nel corso della sessione). h e k rappresentano le percentuali di ogni sottocomponente rispetto al punteggio base totale.

- **EF**: rappresenta la componente di **efficienza** e serve sostanzialmente ad evitare che un tester cerchi di aumentare eccessivamente il proprio punteggio cliccando ripetutamente su widget già evidenziati in precedenza. Viene dunque calcolata come:

$$EF = \frac{w_{hl}}{w_{int}}$$

dove w_{hl} è il numero totale di *highlighted widget* della sessione, mentre w_{int} è il numero di interazioni effettuate sui widget (comprensivo quindi dei click effettuati su widget già evidenziati). Per la natura stessa di questa componente, se il test viene effettuato coscienziosamente, il suo valore sarà tendenzialmente molto vicino a 1.

La somma di queste tre componenti, moltiplicate per i rispettivi coefficienti, permette al tester di incrementare il proprio punteggio fino ad un massimo di 100 punti. Inoltre, per aiutarlo nella comprensione della qualità della propria performance, valutata sulla base del punteggio ottenuto, il plugin fornisce anche un **voto** letterale secondo un meccanismo comunemente utilizzato in molti videogiochi, seguendo una scala con voti che vanno dalla D (voto più basso) alla S (voto più alto), secondo la suddivisione mostrata in Tabella 2.1.

Tabella 2.1. Voti basati sul punteggio finale

Minimo	Massimo	Voto
100	-	S
80	99	A
70	79	B
50	69	C
0	49	D

Un tester ha la possibilità di accumulare anche un **punteggio bonus**, che può arrivare al massimo fino al 50% del punteggio base ottenuto. Il punteggio bonus viene sommato a quello base per determinare la fascia di voto del tester, permettendo quindi di incrementare il voto che si sarebbe ottenuto con il solo punteggio base. In certi casi, dunque, il punteggio bonus permette anche di superare i 100 punti massimi ottenibili grazie al punteggio base.

Il punteggio bonus si ottiene con la formula:

$$\mathbf{P_{bonus}} = [\mathbf{d} \cdot \mathbf{T}] + [\mathbf{e} \cdot \mathbf{PR}]$$

dove:

- **T**: rappresenta la componente **temporale** legata alla durata della sessione. In generale, il plugin vuole premiare chi effettua sessioni più lunghe e quindi (teoricamente) più approfondite, ma per escludere casi limite in cui un tester ha volutamente trascorso troppo tempo sulle pagine per aumentare il proprio punteggio oppure per individuare chi ha cliccato troppo velocemente sui widget soltanto per incrementare il proprio numero di interazioni, viene preso in considerazione anche il tempo medio impiegato per ciascuna interazione. La componente viene dunque calcolata come:

$$T = \begin{cases} 0 & s_{int} \leq 2 \ \vee \ s_{int} > 30 \\ 1.5 \cdot t & 2 < s_{int} \leq 5 \\ t & 5 < s_{int} \leq 15 \\ 0.5 \cdot t & 15 < s_{int} \leq 30 \end{cases}$$

  dove t è la durata della sessione calcolata in minuti e s_{int} i secondi medi per interazione.

- **PR**: rappresenta la componente legata alle **problematiche** trovate dal tester nel corso della sessione. Viene calcolata con la seguente formula:

$$PR = \frac{bug + issue + easteregg}{e}$$

$$bug = \begin{cases} x & n_b \geq \frac{x}{\alpha} \\ \alpha \cdot n_b & n_b < \frac{x}{\alpha} \end{cases}$$

$$issue = \begin{cases} y & n_i \geq \frac{y}{\beta} \\ \beta \cdot n_i & n_i < \frac{y}{\beta} \end{cases}$$

$$easteregg = \begin{cases} z & n_{ee} \geq \frac{z}{\gamma} \\ \gamma \cdot n_{ee} & n_{ee} < \frac{z}{\gamma} \end{cases}$$

$$\text{con} \quad x + y + z = e$$

dove *bug*, *issue* e *easter egg* sono le funzioni per il calcolo delle corrispondenti sottocomponenti. Per ognuna di esse sono definiti dei coefficienti di saturazione: valori massimi che possono assumere le funzioni (rispettivamente x, y e z). Oltre ai coefficienti di saturazione, vengono definiti anche i coefficienti α, β e γ, che rappresentano il peso che ogni unità di ciascuna problematica ha rispetto alla propria sottocomponente. I restanti termini, invece, rappresentano il numero assoluto di ciascun problema: n_i è il numero di issue aperte dal tester, n_{ee} quello di easter egg trovati e n_b il numero di bug riscontrati nel corso della sessione.

Sebbene i pesi di questi addendi siano interamente personalizzabili, come tutti gli altri coefficienti presenti nelle varie componenti del punteggio, è importante dare più rilevanza alla componente legata agli easter egg, poiché essa, in molti casi, si rivela un buon indicatore della profondità con cui è stata svolta la sessione di testing dall'utente corrente. Per tale motivo, di default, al coefficiente γ viene assegnato il triplo del valore degli altri coefficienti.

Questa componente viene inclusa nel punteggio bonus anziché in quello base poiché potrebbero verificarsi dei casi in cui l'interfaccia soggetta al test non presenti alcuna problematica e ciò non dovrebbe inficiare in nessun modo la performance del tester.

Le componenti appena descritte vengono moltiplicate per i rispettivi coefficienti *a, b, c, d* ed *e*, tutti interamente modificabili tramite l'apposito file di configurazione. Nonostante ciò, vengono proposti i seguenti valori di default, che vengono inoltre attribuiti ai coefficienti nel caso in cui i valori presenti nel file di configurazione non rispettassero i vincoli:

- $a = 0.6$;
- $b = k + h = 0.1 + 0.2 = 0.3$;
- $c = 0.1$;

- $d = 0.25$;
- $e = x + y + z = 0.05 + 0.05 + 0.15 = 0.25$;

Al termine di una sessione di testing, il plugin mostra al tester le statistiche ad essa relative all'interno di una finestra di recap, illustrata in Figura 2.7. Tale schermata mostra un riepilogo strutturato delle metriche più importanti che contribuiscono al calcolo del punteggio finale, anch'esso indicato al centro della finestra, distinguendo fra punteggio base e punteggio bonus. Al fondo del riepilogo, inoltre, viene mostrato il voto assegnato alla sessione, sulla base del punteggio totale ottenuto dal tester.

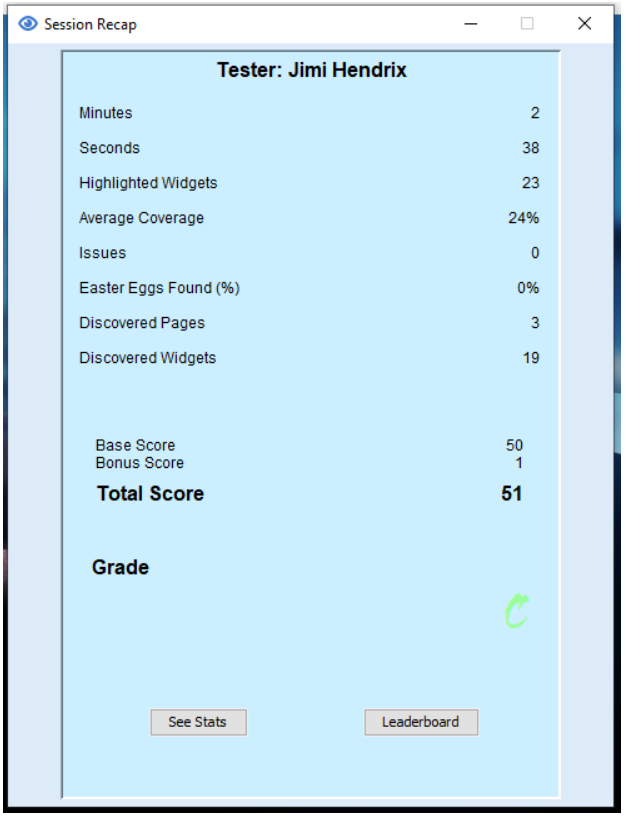

Figura 2.7. Cattura della schermata di riepilogo, contenente tutte le metriche relative alla sessione appena conclusa.

Tramite l'apposito tasto (*See Stats*) presente al fondo della schermata di riepilogo, il tester ha inoltre la possibilità di visualizzare lo storico delle proprie statistiche complessive, aggiornato alla sessione appena conclusa. Tale schermata, rappresentata in figura 2.8, mostra sostanzialmente l'entry del database relativa al tester che

ha svolto la sessione di testing appena conclusa, aggregando tutte le metriche delle sessioni da lui effettuate precedentemente. Tale entry consiste in un oggetto di classe *Stats* ritornato dal metodo *computeStats()* di *statsComputer*, che viene parsificato e passato a sua volta come parametro ad una classe specifica che produce la GUI della schermata di riepilogo.

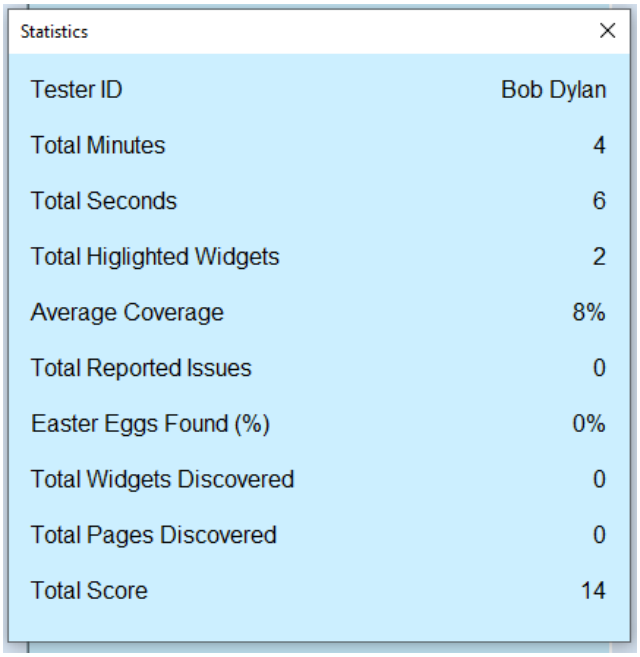

Figura 2.8. Cattura della schermata contenente lo storico relativo ad un tester.

2.7.3 Classifica

Al termine di una sessione, il tester, premendo l'apposito tasto (*Leaderboard*) al fondo della schermata di riepilogo, ha la possibilità di visualizzare una classifica degli utenti che hanno totalizzato il maggior numero di punti nel corso delle varie sessioni. Tale classifica, mostrata in Figura 2.9, mostra i dieci migliori tester che hanno effettuato almeno una sessione di testing, ordinati per punteggio complessivo.

Questo elemento di Gamification è particolarmente importante poiché introduce una componente competitiva all'interno del tool: ogni tester, infatti, potrebbe sentirsi motivato ad effettuare più sessioni per accumulare più punti possibili e superare i propri colleghi in classifica.

L'introduzione di questo elemento apre la strada a scenari interessanti nell'ambito di sessioni di testing svolte in un contesto controllato: per esempio, si può stabilire un certo periodo di validità della classifica e premiare, con una ricompensa reale, il tester che al termine del periodo prestabilito ha totalizzato il maggior numero di punti. Inoltre, andando per esempio a limitare il numero di sessioni effettuabili da ciascun tester, la classifica potrebbe portare ad un incremento dell'efficienza dei tester, incentivandoli a svolgere sessioni di maggiore qualità.

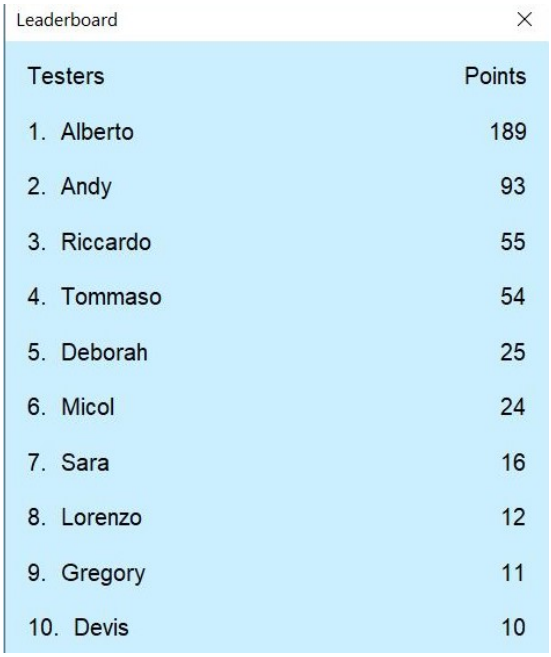

Figura 2.9. Cattura della schermata contenente la classifica dei dieci tester che hanno totalizzato il maggior numero di punti nel corso delle varie sessioni di testing.

2.7.4 Easter Egg

In una sessione di testing su un dominio noto, si deve verificare che l'applicazione risponda agli input ricevuti modificando il proprio stato in modo consono e producendo un particolare output. In un sito web, oltre all'interazione con gli elementi di input, è necessario testare i vari link che compongono la pagina. Tra questi, non è infrequente che alcuni diventino obsoleti, rendendo così la pagina affetta da bug.

Un link in una pagina è detto *broken* (o *corrotto*) quando non individua più la risorsa cui originariamente faceva da puntatore. Questo tipo di bug non è facilmente individuabile a priori, poiché non è noto anticipatamente se il collegamento tra

due pagine sia funzionante finché non lo si ha percorso. Inoltre, tale bug potrebbe esistere solo temporaneamente, nel caso in cui qualche risorsa non fosse raggiungibile per errori di rete, sovraccarico o altri disservizi molto comuni in una rete best effort quale Internet.

Il plugin di Gamification, per simulare la presenza di link corrotti, introduce il concetto di easter egg come un elemento visivo esterno all'applicazione di forma di ovale e dimensioni contenute (inscritto in un rettangolo di 30x50 pixel) presente in sovraimpressione in pagine casuali. Un esempio di easter egg è visibile in Figura 2.10.

Ogni volta che viene aperta una pagina, a partire dalla radice viene estratto a sorte uno tra gli elementi cliccabili contenenti collegamenti a pagine diverse. Se l'elemento scelto viene cliccato, al caricamento della nuova pagina l'Augmented Layer inserisce un ovale colorato in un punto casuale della schermata. Per trovare l'easter egg, il tester è quindi incoraggiato ad esplorare il dominio aprendo il maggior numero di link possibili. Ciò, teoricamente, dovrebbe indurre il tester a condurre una sessione con una coverage mediamente più alta, proprio perché mettendosi alla ricerca dell'easter egg vengono prodotte un numero maggiore di interazioni.

Figura 2.10. Cattura della schermata in cui è presente l'easter egg rappresentato come un ovale giallo.

2.7.5 Evidenziazione delle pagine scoperte

All'interno del plugin, è stato introdotto un altro elemento volto ad incentivare i tester a visitare più approfonditamente un determinato dominio. Tale elemento consiste in un meccanismo che va a marcare una pagina del dominio che non era ancora stata scoperta fino a quel momento (cioè mai visitata prima d'ora da un tester precedente) con una stella gialla semitrasparente posta nell'angolo in alto a sinistra della pagina.

A differenza dei badge e delle ricompense, che vengono spesso utilizzate per testimoniare il raggiungimento di un obiettivo o certificare l'abilità del tester e sono dei riconoscimenti persistenti che una volta raggiunti non possono essere revocati, tale elemento è una via di mezzo tra un premio ed un feedback visivo che al termine della sessione non sarà più visibile al tester. Nonostante ciò, il numero totale di pagine scoperte durante la sessione sarà visualizzabile all'interno della schermata di riepilogo finale.

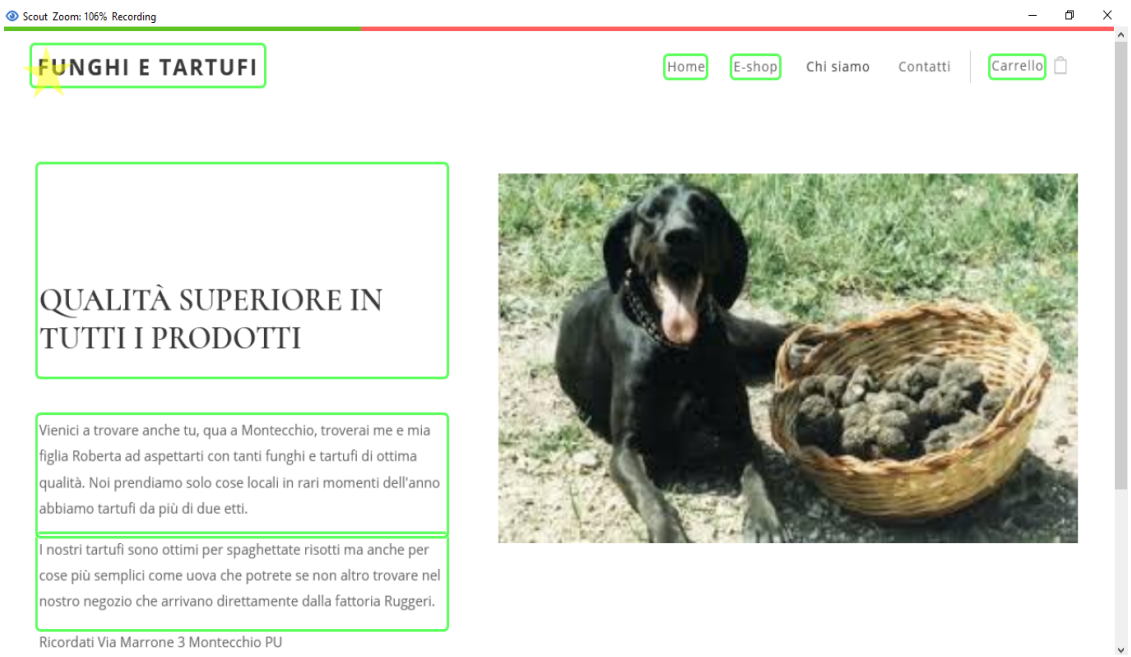

Figura 2.11. Una pagina mai stata visitata, contrassegnata dalla stella gialla in trasparenza.

2.7.6 Elementi teorizzati

Un possibile elemento di Gamification applicabile nel plugin consiste nell'introduzione di meccanismi di sfida. Tale elemento è solo teorizzato e lasciato a possibili ulteriori sviluppi della presente tesi. Le sfide si dividono in due tipi: le sfide esplorative su un domino e le sfide di velocità su una pagina o un insieme di pagine.

Le sfide esplorative consistono nella ricerca del maggior numero di easter egg possibili e nella visita del maggior numero di pagine possibili. Il tester potrà selezionare un home locator tra un pool di possibili siti da testare o inserirne uno a sua scelta, purché l'URI inserito conduca all'home page di un dominio. È importante che venga specificata la pagina iniziale poiché verranno considerate valide ed esplorate solo le pagine contenenti nell'URL la sottostringa dell'home locator.

Una volta selezionato il dominio di testing, il tester avrà la libertà di esplorare quante più pagine possibile senza essere cronometrato. Per vincere la sfida dunque bisognerà superare l'attuale record di pagine esplorate per dominio o di eguagliare il record, ma trovando un numero maggiore di easter egg durante la sessione di testing. Non è quindi necessario interagire con tutti i widget presenti nelle varie pagine, ma è richiesta solo l'interazione con quelli in cui risiedono i collegamenti tra le varie pagine.

Le sfide di velocità su un insieme di pagine consistono nell'ottenimento della massima coverage di sessione possibile sull'insieme prestabilito. Il tester partecipa alla sfida scegliendo nuovamente l'home locator tra un pool di siti esistenti o aggiungendone uno al pool seguendo gli stessi criteri della sfida esplorativa.

Il tester dovrà quindi essere il più veloce ad interagire su tutti i widget nel minor tempo possibile. A parità di coverage sarà infatti il miglior tempo a decretare il vincitore della sfida.

Quando un tester risulta essere il campione di un dominio su cui è attiva una o più sfide, viene mostrato il trofeo di vittoria ogni volta che si esegue una sessione di testing su quel particolare dominio. Entrambe le sfide possono essere facilmente resettate, nel caso in cui ad esempio il dominio di testing venisse aggiornato ad una nuova release o su base periodica (ogni trimestre/semestre).

Un altro possibile elemento di Gamification consiste nell'implementazione degli achievement, ossia di particolari riconoscimenti permanenti che ogni tester può guadagnare una volta raggiunti determinati obiettivi.

Ogni achievement generalmente ha un nome particolare che descrive, spesso anche simpaticamente, l'obiettivo da raggiungere (che può anche essere nascosto, nel caso degli achievement più rari), ossia la missione da compiere per ottenere l'achievement stesso. Una volta conquistato un achievement, esso sarà visualizzabile permanentemente nell'apposita sezione del profilo del tester, dove sarà quindi possibile tenere traccia degli achievement ottenuti e di quelli non ancora guadagnati.

Solitamente gli achievement sono uguali per tutti i tester che utilizzano il tool e sono in numero prestabilito. Ciò comporta che sia possibile realizzare una graduatoria degli utenti in base al numero di achievement collezionati, oltre al fatto di poter classificare gli achievement stessi con una scala di rarità in base alla percentuale di utenti che hanno raggiunto ciascun obiettivo.

Ci possono essere diverse tipologie di achievement, spaziando da quelli introduttivi (es. "Chi ben comincia..": "Hai effettuato la tua prima sessione di testing."), che cercano di fidelizzare gli utenti ricompensandoli fin dalle prime fasi di utilizzo della piattaforma così da invogliarli a tornare, fino a quelli cumulativi, più comuni, che premiano particolari traguardi come la ripetizione di una determinata operazione un certo numero di volte (es. Achievement "Stakanovista!": "Hai effettuato 100 sessioni di testing."). Altri ancora, invece possono corrispondere a piccole sfide isolate (es. "Gallina dalle uova d'oro": "Hai trovato tutti gli easter egg presenti in una singola sessione di testing."), con cui si vuole premiare un tester che riesca a far accadere un certo evento in un dato intervallo di tempo.

Gli achievement sono strettamente legati ad un altro elemento di Gamification facilmente implementabile, ossia i badge. I badge non sono altro che coccarde virtuali ottenibili da ciascun tester al raggiungimento di determinati obiettivi e ciascuno di essi certifica una particolare competenza acquisita dal tester.

I badge sono dunque legati ad obiettivi meno specifici rispetto a quelli a cui fanno riferimento gli achievement, sebbene siano anch'essi legati al raggiungimento di determinati traguardi. Presentano, inoltre, altre due differenze sostanziali:

1. Ogni badge corrisponde ad una coccarda con un aspetto unico, visualizzabile immeditamente all'interno del profilo di ciascun tester. Ciò può spingere gli utilizzatori del plugin a voler collezionare più badge possibili anche per migliorare esteticamente il proprio profilo personale.

2. Dato che ogni badge certifica una determinata competenza acquisita dal tester, è possibile esportarli in un file esterno al plugin in modo da poter attestare, anche al di fuori del contesto del tool, l'avvenuta acquisizione di tali conoscenze.

Capitolo 3

Esempio di sessione

Grazie alle classi descritte nel dettaglio nella Sezione 2.3, il plugin permette di creare una rappresentazione della sessione di testing effettuata dal tester corrente, creando un albero delle pagine visitate.

Per illustrare il funzionamento di questo meccanismo di rappresentazione, si è scelto di effettuare una semplice sessione di testing su un sito web di tipo marketplace[1], che si presta adeguatamente alla realizzazione di un esempio di utilizzo del plugin per via della semplicità delle sue pagine.

La sessione è stata svolta seguendo i passi qui elencati:

1. All'interno della pagina principale del sito, il cui indirizzo è stato indicato come *home locator* all'interno di Scout), è stato cliccato il widget presente nell'header della pagina per spostarsi all'area del negozio;

2. All'interno del negozio virtuale, è stato selezionato l'articolo "Porcini";

3. Nella pagina di acquisto dell'articolo selezionato precedentemente, è stata utilizzata la scrollbar per visualizzare il footer della webpage e si è cliccato il link diretto alla pagina riguardante l'informativa sulla privacy;

4. Una volta raggiunta la pagina dedicata alla privacy, è stato utilizzato l'apposito tasto *Go Home*, messo a disposizione da Scout all'interno della propria toolbar, per tornare all'home locator, ossia la pagina principale del sito;

[1] https://magi-giovanni-funghi-e-tartufi.webnode.it/

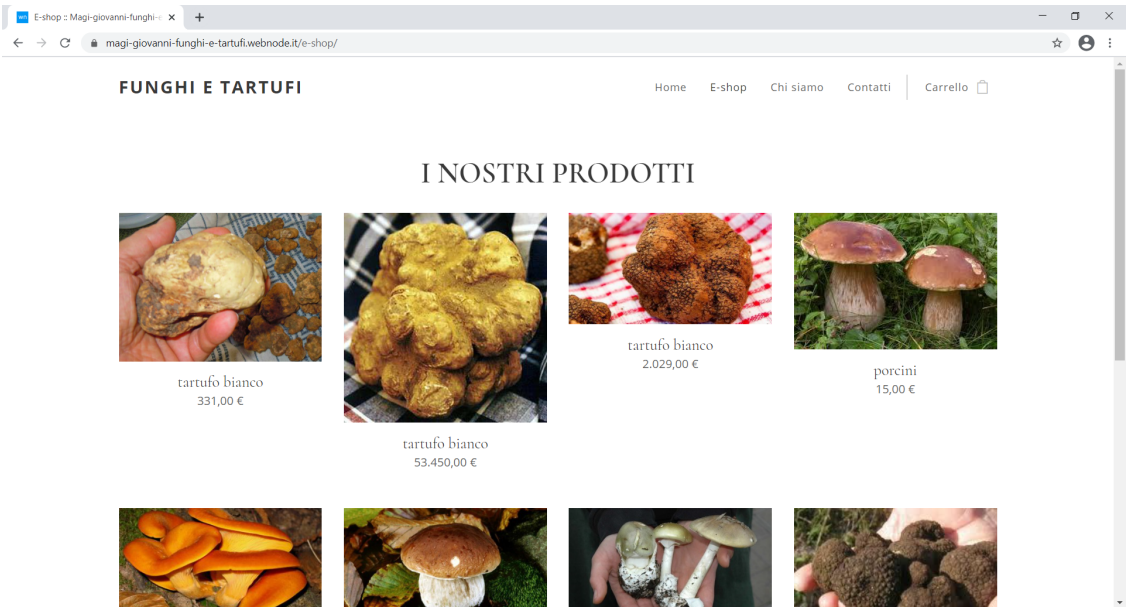

Figura 3.1. E-shop del sito scelto per l'esempio.

5. Dopo aver raggiunto nuovamente l'homepage, è stato cliccato il link relativo alla pagina "Chi Siamo";

6. All'interno della pagina "Chi Siamo", si è scelto nuovamente di cliccare un link presente all'interno dell'header, per raggiungere l'area contenente i contatti del venditore;

7. Una volta raggiunta la pagina dei contatti, è stato cliccato il link, sempre presente nell'header, per tornare nell'area del negozio virtuale;

8. All'interno della pagina del catalogo appena raggiunta, è stato utilizzato il tasto *Go Home* per fare ritorno all'home locator;

9. All'interno dell'homepage, è stato cliccato nuovamente il widget per raggiungere l'e-shop;

10. Nella pagina del negozio virtuale è stato scelto l'articolo "Tartufo Bianco" dal catalogo;

11. Una volta raggiunta la pagina per la selezione della quantità di articoli del tipo scelto da acquistare, è stato cliccato il link presente nell'header relativo al "Carrello" dell'utente;

12. All'interno della pagina del carrello, si è scelto di terminare la sessione premento il tasto *Stop* presente nella toolbar di Scout.

Questa sequenza di operazioni si può visualizzare graficamente con l'albero della sessione presente in Figura 3.2. Si noti che le visite multiple di una stessa pagina non sono presenti all'interno dell'albero poiché visitare nuovamente una pagina su cui si è già stati non porta alla creazione di un nuovo ramo dell'albero.

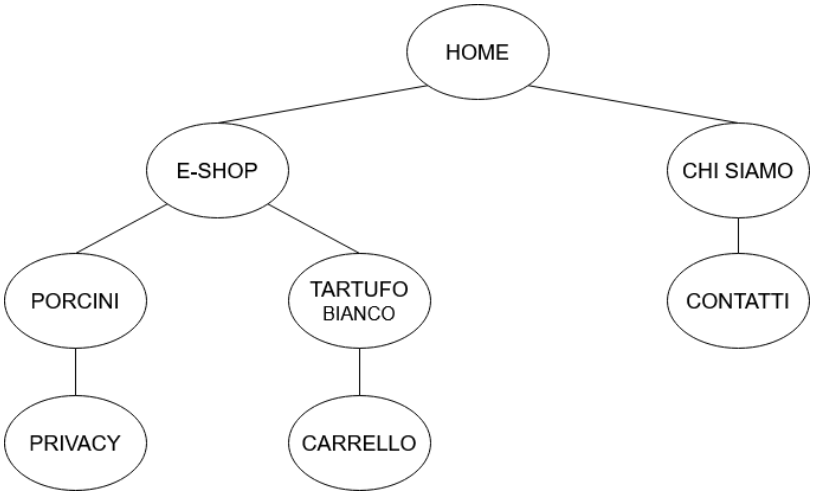

Figura 3.2. Rappresentazione grafica dell'albero della sessione.

È possibile ottenere un riepilogo delle pagine visitate durante la sessione anche nella forma di output testuale all'interno del terminale, invocando il metodo *printTree()* fornito dalla classe *Session*, che in questo caso specifico produrrebbe l'output presente in Figura 3.3. Tale output si ottiene effettuando una visita ricorsiva *pre-order* dell'albero ed evidenzia il livello di profondità di ciascuna pagina utilizzando un simbolo "—>" per ciascun livello superiore a quello di partenza (a cui corrisponde soltanto il nodo relativo all'home locator).

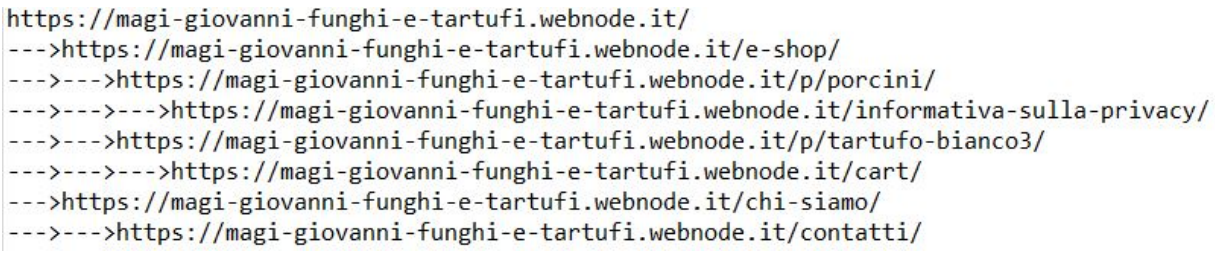

Figura 3.3. Output della visita pre-order dell'albero della sessione.

Capitolo 4

Validazione del plugin

Terminata la fase di teorizzazione e sviluppo del plugin, è stata avviata la fase di sperimentazione su un campione ristretto di dieci persone, al fine di validare l'efficacia del lavoro svolto. Si noti che la ristrettezza del campione è stata fortemente influenzata dalle restrizioni dovute alla pandemia di Covid-19, che non hanno reso possibile l'accesso a laboratori universitari o ad altri luoghi pubblici che permettessero di effettuare l'esperimento in presenza e di avere un numero più nutrito di partecipanti. Per tutte queste ragioni, la validazione del plugin è stata eseguita a distanza, fornendo a ciascun partecipante tutto il necessario per svolgere in autonomia l'esperimento (che non si è dunque potuto svolgere sotto diretta osservazione).

4.1 Selezione del campione

Il tool all'interno del quale è stato implementato il plugin di Gamification è uno strumento professionale di GUI testing, quindi pensato per essere utilizzato da persone con un nutrito background di sviluppo software e di testing. Per tale ragione, i partecipanti all'esperimento sono stati scelti da un ambiente molto simile al target reale del tool. I soggetti sono stati per lo più studenti del corso di Laurea Magistrale in Ingegneria Informatica o in Informatica. Una piccola parte del campione è invece composta da lavoratori nel mondo dell'informatica, con una Laurea Triennale o Magistrale presso le facoltà sopra citate. L'età del campione varia tra i 18 e i 30 anni. Ciò influisce sull'esperienza ed il background posseduto dai partecipanti: come si evince dai grafici mostrati in Figura 4.1, Figura 4.2 e Figura 4.3 (che mostrano

rispettivamente le risposte alle domande 1.4, 1.5 e 1.8 del questionario in Tabella 4.1), la maggior parte dei partecipanti ha avuto esperienze precedenti di programmazione web, ma soltanto due di loro avevano effettuato delle sessioni di testing prima di utilizzare Scout in occasione dell'esperimento. Si noti che la mancanza di esperienza nell'ambito del testing non rappresenta un elemento particolarmente influente, in quanto il tipo di testing oggetto della sperimentazione è principalmente esplorativo e prevede l'esecuzione di azioni manuali e intuitive, poiché molto simili a quelle svolte tipicamente durante una normale visita di un sito web.

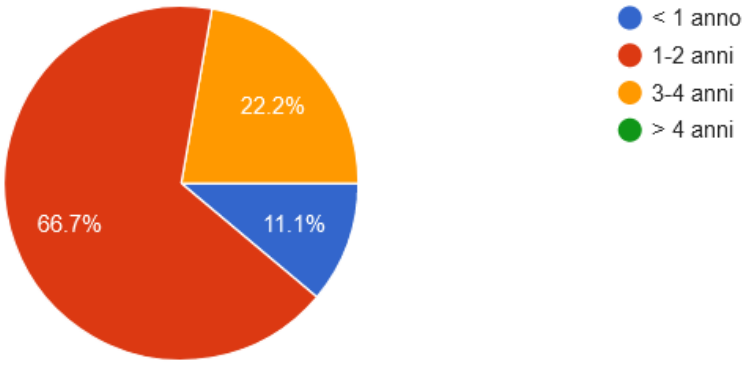

Figura 4.1. Anni di esperienza dei partecipanti nella programmazione web.

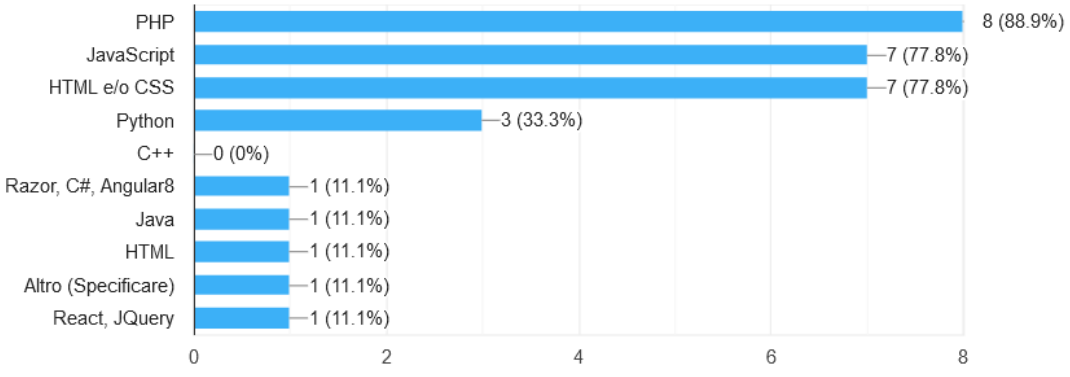

Figura 4.2. Linguaggi utilizzati dai partecipanti per la programmazione web.

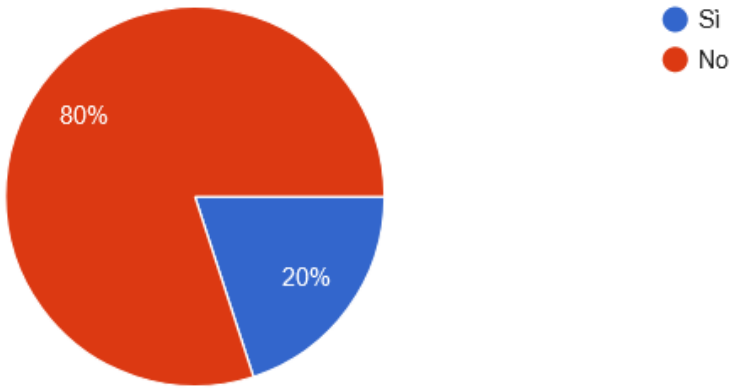

Figura 4.3. Anni di esperienza dei partecipanti nel testing di applicazioni web.

4.2 Metodologia

4.2.1 Descrizione dell'esperimento

Per valutare il lavoro si è scelto uno schema di valutazione AB-BA dividendo il campione in due gruppi, nei quali ogni partecipante avesse la possibilità di utilizzare il tool due volte, ognuna su una diversa applicazione: la prima volta utilizzando la versione di Scout originale, la seconda volta sfruttando la versione ludicizzata del tool. L'idea dietro l'adozione di questo schema consiste nel voler evitare che l'esperienza accumulata nella sessione sul primo applicativo possa influenzare in qualche modo l'esperienza ludicizzata.

Le applicazioni utilizzate per le due sessioni di testing sono state:

- Applicazione A: sito di tipo marketplace[1];
- Applicazione B: sito di compravendita online[2];

Il primo gruppo ha svolto la prima sessione sull'applicazione A e la seconda sull'applicazione B, mentre il secondo gruppo ha invertito i siti da testare. Utilizzando

[1]https://magi-giovanni-funghi-e-tartufi.webnode.it/

[2]https://www.subito.it/

questo schema, ogni soggetto è stato messo nelle condizioni di comprendere la differenza tra l'utilizzo dello stesso strumento con e senza Gamification e di esprimere la propria opinione sull'impatto degli elementi ludici sull'esperienza complessiva, sia in positivo che in negativo. Lo schema riassuntivo dell'esperimento è rappresentato in Figura 4.4.

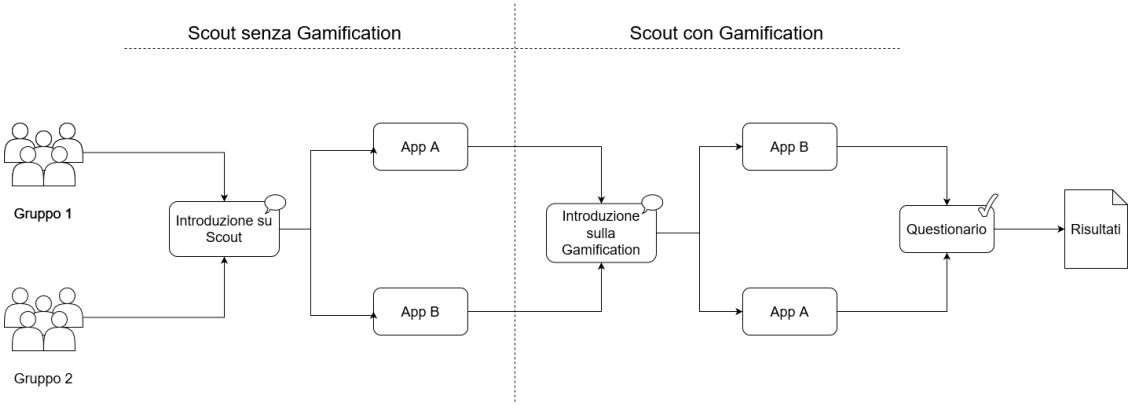

Figura 4.4. Organizzazione dell'esperimento.

A tutti i partecipanti è stato fatto un discorso introduttivo sull'utilizzo di Scout mediante uno script testuale fornito con il resto del materiale necessario per l'esecuzione dell'esperimento, in modo che fin dalla prima sessione fossero noti i meccanismi alla base dell'Augmented Testing e, in particolare, del funzionamento di Scout stesso. Successivamente, si è lasciato libero il soggetto di effettuare una sessione di testing esplorativo, della durata massima di quindici minuti, all'interno della prima applicazione.

Alla prima sessione è seguita un'altra spiegazione testuale riguardante la Gamification e gli elementi ludici presenti nel tool una volta attivato il plugin, dei quali sono stati resi noti sia l'aspetto che il funzionamento, al fine di rendere i tester consapevoli riguardo al motivo dietro il loro inserimento.

La seconda sessione, infine, è stata eseguita dai tester sulla seconda applicazione e con il plugin attivo, anch'essa della durata massima di 15 minuti.

Si noti inoltre che, nonostante gli elementi visivi di Gamification siano attivi solo nella seconda fase dell'esperimento, il tool è stato opportunamente modificato in modo da produrre le metriche di interesse anche durante lo svolgimento della prima parte. Ciò ha permesso di effettuare un confronto delle effettive performance dei tester basato su dati oggettivi e non sulla sola percezione dell'utente.

La fase finale della sperimentazione è stata la compilazione del questionario mostrato in Tabella 4.1 tramite un Google Form in cui, dopo alcune domande introduttive sui dati anagrafici e sul background posseduto, è stata chiesta l'opinione del partecipante riguardo alla percezione dell'inserimento nel tool degli elementi ludici presentati in precedenza. È stato lasciato inoltre spazio alla segnalazione di eventuali problematiche ed imprevisti riscontrati, insieme ad uno spazio adibito all'inserimento di consigli e suggerimenti. È stato infine richiesto a ciascun tester di comprimere i file contenenti le metriche relative alle proprie sessioni all'interno di un archivio, caricato come allegato al termine del form.

Tabella 4.1: Domande del Questionario

ID	**Domanda (Tipo)**
1.1	Età (Scelta Multipla)
1.2	Hai mai avuto esperienze professionali con Java, come studente o lavoratore? (Scelta Multipla)
1.3	Quanti anni di esperienza hai con la programmazione in Java? (Scelta Multipla)
1.4	Quanti anni di esperienza hai con la programmazione Web? (Scelta Multipla)
1.5	Quali tra questi linguaggi hai mai usato per la programmazione Web? (Checkbox)
1.6	Hai mai avuto esperienze di testing di applicazioni Java? (Scelta Multipla)
1.7	Quali tool hai utilizzato per fare testing di applicazioni Java? (Aperta)
1.8	Hai mai avuto esperienze di testing di applicazioni o siti Web? (Scelta Multipla)
1.9	Quali tool hai utilizzato per fare testing di applicazioni Web? (Aperta)
2.1	Hai compreso il modo di utilizzare Scout e il suo contesto di utilizzo? (Likert)

Tabella 4.1: Domande del Questionario

ID	Domanda (Tipo)
2.2	Hai trovato la Barra di Progresso utile a mostrare i tuoi progressi nel testing di una pagina? (Likert)
2.3	La Classifica finale ha stimolato il tuo senso di sfida verso i tester precedenti? (Likert)
2.4	Conoscere l'esistenza di un Punteggio finale ti ha spinto a migliorare la tua performance? (Likert)
2.5	La Stella che contrassegna le pagine scoperte ti ha incoraggiato ad esplorare più approfonditamente l'applicazione da testare? (Likert)
2.6	Gli Easter Egg indotti ti hanno stimolato ad interagire con il maggior numero possibile di widget all'interno dell'applicazione? (Likert)
2.7	Quali tra i seguenti elementi erano a te familiari prima dello svolgimento dell'esperimento? (Checkbox)
2.8	Quali, tra gli elementi selezionati nella risposta alla domanda precedente, sono stati utilizzati in maniera a te familiare? (Checkbox)
2.9	Quali tra i seguenti elementi ritieni essenziali in una sessione di testing? (Checkbox)
2.10	Hai trovato comprensibile la Schermata di Riepilogo e facilmente decodificabili le informazioni presenti al suo interno? (Likert)
2.11	Trovi che il tool utilizzato sia facilmente integrabile in un ambiente di testing esistente (lavorativo o di studio)? (Likert)
2.12	La versione di Scout che utilizza la Gamification ha aumentato la tua consapevolezza riguardo alla tua performance rispetto a quella senza? (Likert)
2.13	Nel complesso, trovi migliore la versione di Scout che utilizza la Gamification rispetto a quella senza? (Likert)
2.14	Hai riscontrato dei problemi durante lo svolgimento dell'esperimento? Se sì, descrivili brevemente (Aperta)
2.15	Hai dei suggerimenti per migliorare il tool proposto? (Aperta)

4.2.2 Preparazione dell'ambiente di svolgimento

L'esperimento è stato svolto a distanza da tutti i partecipanti, utilizzando una macchina virtuale con sistema operativo Windows 10 Home resa disponibile direttamente dal sito di Microsoft per un periodo di prova limitato. È stato inoltre chiesto ad ogni tester di effettuare il download dell'ultima versione disponibile di VirtualBox (6.1) e di importare la VM descritta in precedenza, anch'essa fornita tramite un apposito link per il download.

Ogni VM è stata corredata di tutto l'occorrente per la corretta esecuzione dell'espermimento e la compilazione del questionario. In particolare, sono state preventivamente installate le ultime versioni di Firefox e di 7-Zip, rispettivamente per poter svolgere le sessioni di testing e per poter generare l'archivio contenente i risultati ottenuti. Infine, è stato fornita ad ogni partecipante una directory contenente tutte le dipendenze ed un file *.jar* per dare la possibilità di avviare Scout in maniera semplice e veloce, senza che fosse necessario investire del tempo in ulteriori configurazioni del tool.

L'utilizzo della VM, oltre ad aver reso possibile lo svolgimento degli esperimenti a distanza, ha permesso a ciascun tester di partire dal medesimo stato di partenza, sia in termini di ambiente (risorse, sistema operativo, memoria), sia per quanto riguarda gli aspetti ludicizzati: ogni partecipante (indipendentemente dal gruppo di appartenenza) si è trovato di fronte ad uno stato di partenza non vuoto, poiché in precedenza sono state effettuate due sessioni campione su ciascuno dei domini da testare, tutte della durata di circa quindici minuti ciascuna. Ciò è stato reso possibile anche grazie alla completa configurabilità del database, che consente di inserire manualmente una serie di dati, come ad esempio alcune pagine del dominio da testare, in modo da generare uno stato iniziale non nullo.

Lo stato di partenza così generato contiene dunque:

- un set di pagine già visitate;
- un set di widget già interagiti;
- la presenza di due risultati in classifica.

La presenza di questo stato di partenza, per i tester, diventa rilevante soltanto durante la sessione in cui viene attivato il plugin di Gamification, ma a livello statistico la sua presenza ha un impatto su entrambe le sessioni.

4.3 Analisi delle performance dei tester

Grazie al meccanismo di generazione dei file di log descritto nella Sezione 2.2, è stato possibile effettuare una raccolta dei dati relativi alle sessioni di testing svolte dai partecipanti, sia nel caso di utilizzo della versione base del tool che in quello con la versione ludicizzata. Ciò ha permesso di confrontare i dati relativi alle due sessioni, evidenziando eventuali differenze quantitative tra i due diversi scenari di utilizzo.

Nelle sottosezioni successive si farà riferimento ai dieci tester che hanno preso parte all'esperimento utilizzando le lettere dalla A alla J. Come spiegato nella Sezione 4.2, i dieci partecipanti sono stati separati in due sottogruppi (Gruppo 1 e Gruppo 2), a seconda dell'ordine con cui hanno testato le due applicazioni: il Gruppo 1 ha testato prima l'applicazione A, mentre il Gruppo 2 l'applicazione B. La suddivisione dei dieci partecipanti è rappresentata in Tabella 4.2.

Tabella 4.2. Suddivisione partecipanti

Partecipante	Gruppo
A	1
B	2
C	2
D	2
E	1
F	2
G	1
H	2
I	1
J	1

4.3.1 Coverage media

Il dato più significativo su cui concentrare la nostra analisi consiste nella coverage media raggiunta dai tester nel corso delle due sessioni, che fornisce un'indicazione quantitativa del grado di esplorazione raggiunto. Le percentuali raggiunte dai dieci partecipanti sono visibili in Figura 4.5.

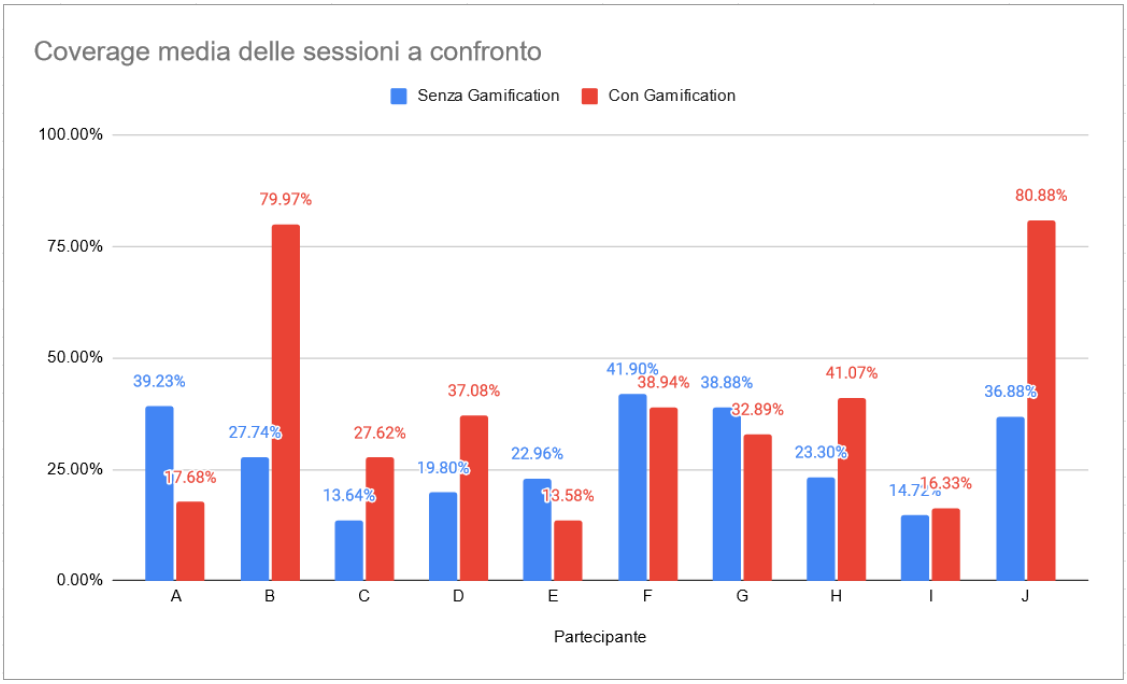

Figura 4.5. Coverage media delle sessioni a confronto.

Come si può notare dal grafico, il plugin di Gamification ha portato ad un incremento della coverage media in sei partecipanti su dieci. È inoltre interessante notare come nella metà di questi casi, la percentuale di coverage raggiunta sia persino raddoppiata rispetto a quella relativa alla sessione non ludicizzata.

È interessante notare anche la variazione di coverage media del partecipante A: esso, infatti, ha mostrato un dimezzamento dovuto principalmente alla maggior complessità del sito sui cui ha effettuato la sessione ludicizzata, oltre che ad una minor durata della seconda sessione (quattro minuti in meno rispetto alla prima).

Calcolando la media del delta di coverage media di ciascun utente, si può notare un incremento del 47%, che evidenzia un generale miglioramento delle performance dei tester nel passaggio dall'utilizzo del tool originale a quello del tool gamificato.

4.3.2 Punteggio

Un altro dato particolarmente significativo, poiché permette di dare una valutazione qualitativa alla performance di un tester, è il punteggio, calcolato con la formula spiegata nel dettaglio all'interno della Sezione 2.7.2. In Figura 4.6, è possibile

osservare i punteggi ottenuti dai tester nelle due sessioni svolte, messi a confronto.

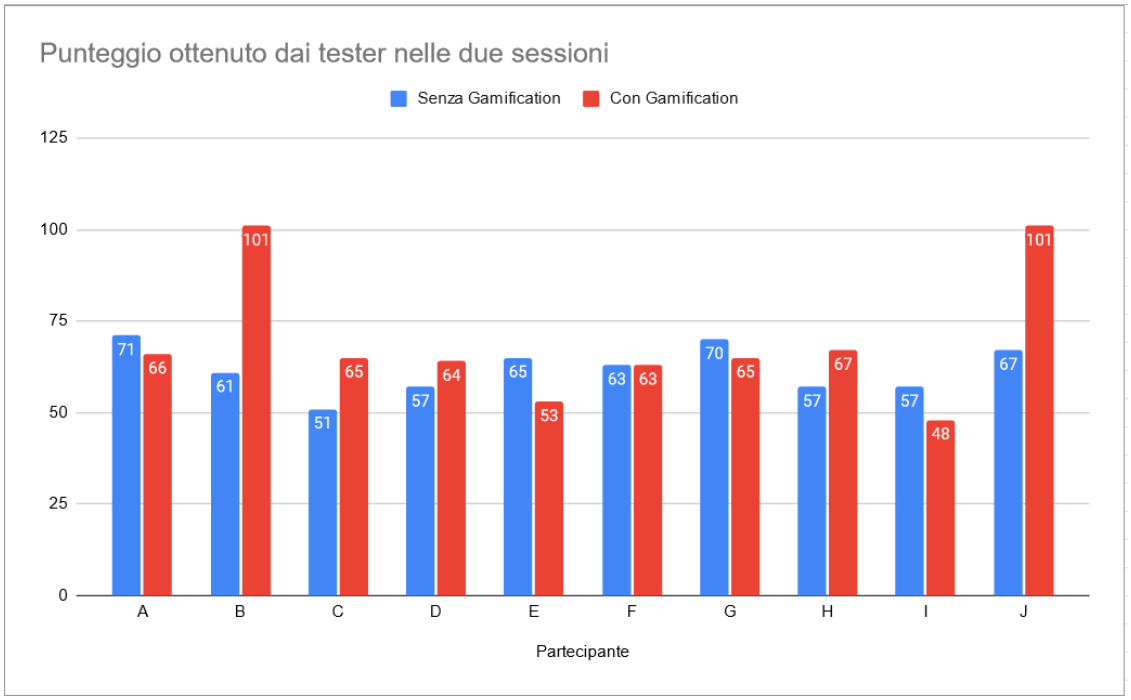

Figura 4.6. Punteggio ottenuto dai tester nelle due sessioni.

Il confronto fra i punteggi ottenuti nelle due sessioni porta ad un risultato che va di pari passo con quello osservato nell'analisi della coverage media: cinque partecipanti hanno incrementato il loro punteggio rispetto alla sessione precedente, mentre uno ha ottenuto il medesimo punteggio in entrambe le sessioni. Questo risultato non sorprende, in quanto la coverage media costituisce la componente più rilevante all'interno del calcolo del punteggio. Nonostante ciò, è interessante notare come soltanto cinque dei tester che hanno incrementato la propria coverage media siano riusciti a convertire questo risultato in un punteggio maggiore nella seconda sessione: il tester I, infatti, ha ottenuto un punteggio inferiore nella sessione ludicizzata, ma è importante sottolineare il fatto che fosse l'unico dei partecipanti sopracitati ad aver ottenuto un incremento di coverage poco rilevante, di circa due punti percentuali.

Tra i tester che, invece, non hanno aumentato la propria coverage media tra una sessione e l'altra, salta all'occhio il partecipante A: esso, infatti, pur avendo dimezzato la propria coverage, è riuscito ad ottenere un punteggio di poco inferiore a quello ottenuto in seguito alla prima sessione. Ciò è dovuto al raddoppiamento

della percentuale di widget nuovi interagiti sui widget totali, rispetto alla prima sessione. Un altro caso particolare è costituito dal partecipante F, che è riuscito ad ottenere lo stesso punteggio nelle due sessioni, compensando il piccolo svantaggio di coverage ottenuto nella seconda sessione grazie alla scoperta di un easter egg e ad una migliore percentuale di pagine nuove scoperte.

4.3.3 Easter Egg

L'ultimo dato quantitativo analizzato consiste nel confronto del numero di easter egg scoperti dai tester, statistica che, come la coverage, fornisce un'indicazione della profondità con cui sono state testate le pagine dei due domini. Il confronto in questione è rappresentato in Figura 4.7.

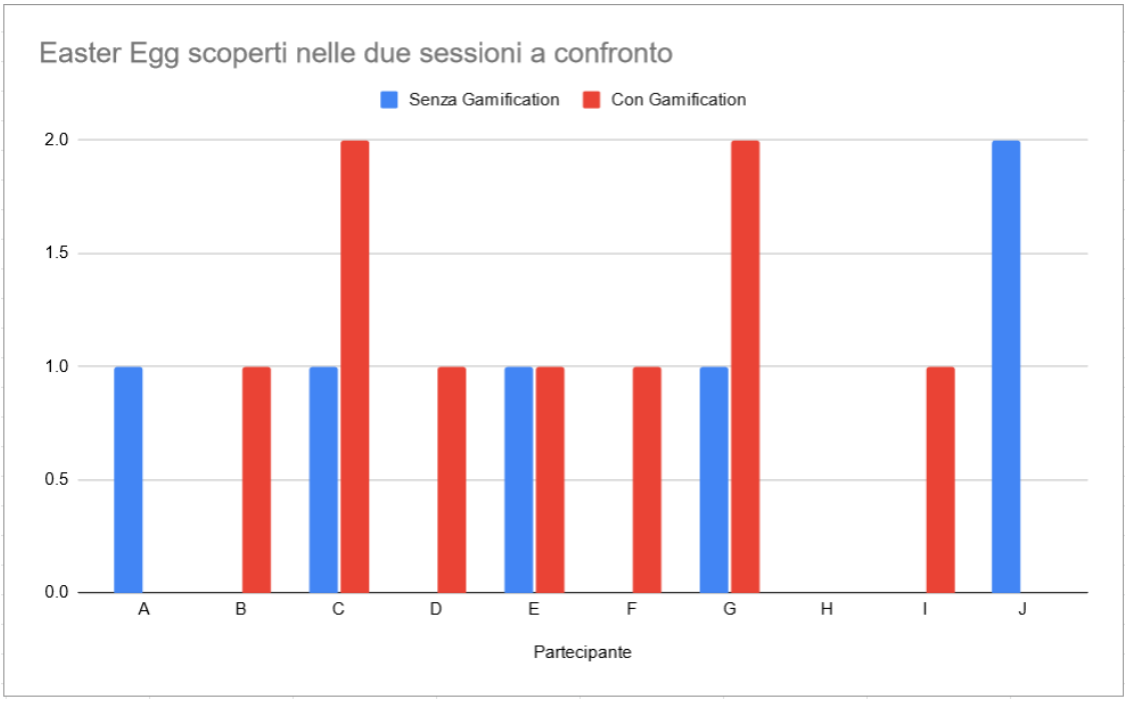

Figura 4.7. Easter Egg scoperti dai tester nelle due sessioni.

Come si può notare dal grafico, con la versione ludicizzata del tool sette partecipanti su dieci hanno trovato almeno un easter egg nel corso della sessione gamificata, mentre solo la metà dei tester avrebbe visualizzato almeno un easter egg nella prima sessione, se il plugin di Gamification fosse stato attivo.

Un altro dato rilevante emerge dal confronto fra le percentuali di easter egg trovate nel corso delle due sessioni, dove ciascuna percentuale viene calcolata come il rapporto fra il numero di easter egg scoperti ed il numero totale di easter egg attivabili. Quest'ultimo corrisponde al numero totale di pagine visitate (escludendo l'homepage), poiché è presente un solo easter egg all'interno di ciascuna pagina. Come si può vedere nel grafico rappresentato in Figura 4.8, sei tester su dieci hanno aumentato la propria percentuale una volta venuti a conoscenza della presenza di questo elemento di Gamification e del relativo feedback visivo.

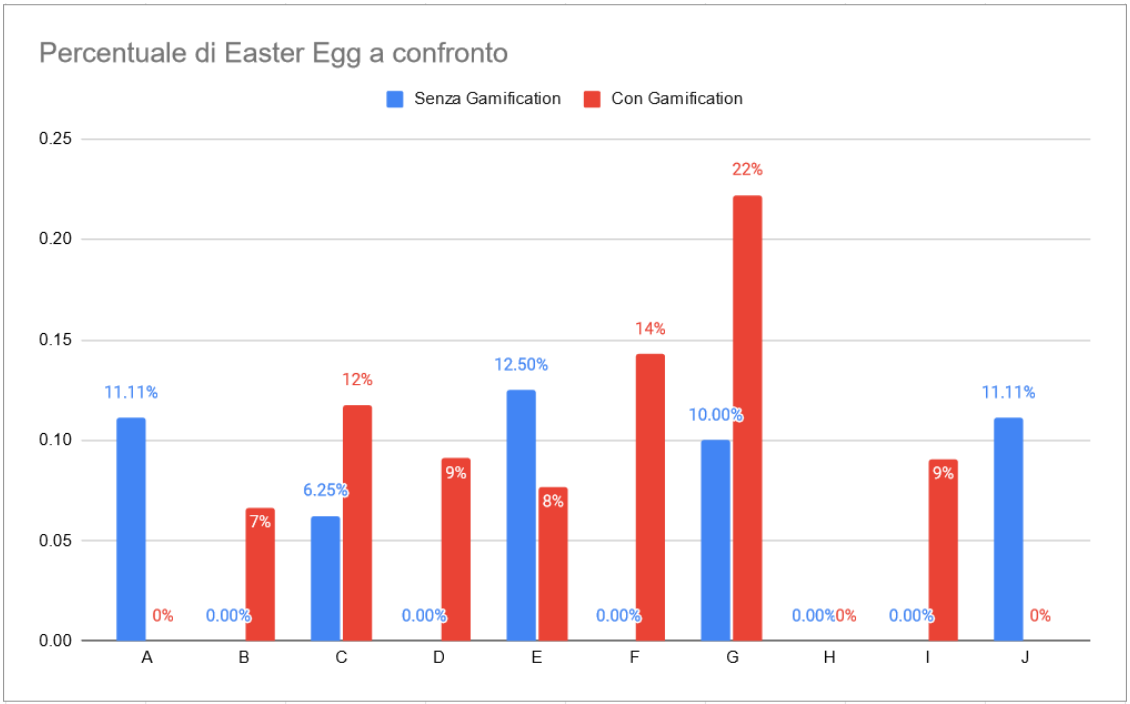

Figura 4.8. Percentuale di easter egg a confronto.

4.4 Risultati del questionario

Come spiegato in precedenza, al termine delle due sessioni di testing è stato chiesto a ciascun partecipante di compilare il questionario presente in Tabella 4.1. Le domande presenti al suo interno, dopo una prima parte focalizzata sui dati anagrafici, sul background culturale e tecnico del tester e sulla comprensione delle spiegazioni riguardanti Scout ricevute prima delle sessioni, si sono concentrate sulla percezione

che ciascun tester ha avuto riguardo agli elementi di Gamification inseriti e il tool nella sua totalità. Ciò ha permesso di formulare delle analisi qualitative dell'impatto di ciascun elemento sull'esperienza di ogni partecipante, le quali, una volta affiancate alle analisi quantitative svolte nella sezione precedente, hanno consentito di ottenere una valutazione più completa sugli aspetti più rilevanti dell'esperimento.

La maggior parte delle domande è stata impostata in modo da richiedere una risposta chiusa che fornisse cinque opzioni, sfruttando la scala Likert: ogni tester aveva la possibilità di selezionare una risposta da 1 ("Decisamente no", risposta fortemente negativa) a 5 ("Decisamente sì", risposta fortemente positiva). Il resto delle domande consentiva invece risposte aperte, ad esempio nel caso di segnalazione di eventuali problematiche (domanda 2.14) o suggerimenti (domanda 2.15).

Il resto della Sezione si focalizzerà sui risultati del questionario relativi a ciascun elemento di Gamification implementato nel tool.

4.4.1 Barra di progresso

La barra di progresso, descritta nell'omonima Sezione 2.7.1, ha l'obiettivo di fornire un feedback visivo al tester per renderlo consapevole dei propri progressi in termini di coverage raggiunta all'interno della pagina corrente. Come risulta dalle risposte alle domande 2.7 e 2.8 del questionario, tutti i partecipanti conoscevano già la barra di progresso come elemento di game design e tutti e dieci hanno trovato che tale elemento sia stato utilizzato in una maniera a loro familiare.

Come si evince dal grafico presente in Figura 4.9, che rappresenta le risposte alla domanda 2.2 del questionario, la maggior parte dei partecipanti ha ritenuto utile la presenza della barra di progresso per rappresentare in tempo reale la coverage raggiunta all'interno della pagina: metà dei tester, infatti, ha assegnato la massima valutazione (5) all'elemento, quattro tester hanno assegnato un voto pari a 4 mentre solo un tester ha ritenuto che l'elemento valesse 3 punti sulla scala Likert.

Nonostante l'efficacia generale percepita dalla maggior parte dei tester, alcuni di loro ritengono che l'aspetto della barra possa essere migliorato, rendendola più evidente e possibilmente animata, oltre che fissandola in cima alla pagina anche in caso di scroll verso il basso.

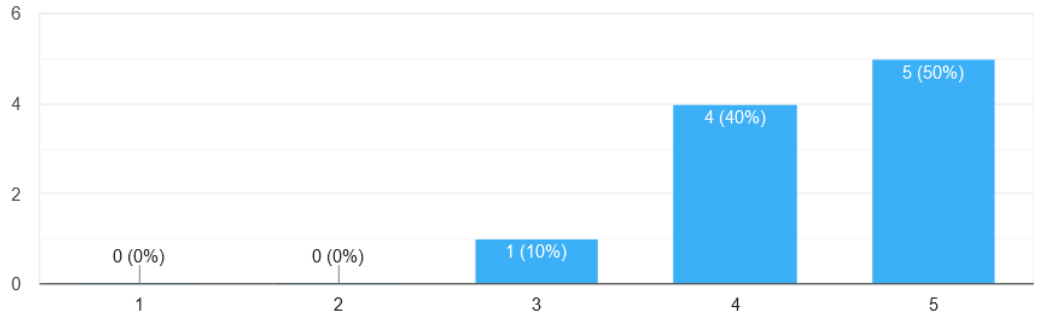

Figura 4.9. Valutazione dell'utilità della barra di progresso nel mostrare i progressi parziali.

4.4.2 Stella

La stella utilizzata come feedback visivo per segnalare la scoperta di pagine non ancora visitate in precedenza da altri tester è risultato l'elemento noto al minor numero di partecipanti: stando alle risposte riguardanti le domande 2.7 e 2.8, infatti, solo quattro tester avevano presente il concetto rappresentato da questo elemento e soltanto due di loro hanno ritenuto che esso sia stato implementato come se lo aspettavano. Ricordiamo che, come descritto nella Sezione 2.7.5, la stella vuole simulare una sorta di premio temporaneo e, tra tutti gli elementi di game design inseriti, è probabilmente quello che ha subito più adattamenti rispetto all'utilizzo tipico, prima di essere implementato all'interno del contesto di Scout.

Nonostante ciò, la percezione avuta dai tester riguardo alla presenza della stella può definirsi generalmente positiva: come mostrato dal grafico in Figura 4.10, più della metà dei partecipanti ha espresso un giudizio positivo o molto positivo, tre tester hanno espresso un giudizio neutrale mentre un solo tester ha valutato molto negativamente l'impatto motivazionale di questo elemento visivo, affermando di non aver notato il feedback per tutta la durata della sessione ludicizzata.

Infine, un partecipante ha suggerito di realizzare un riquadro apposito per mostrare l'eventuale conseguimento di una stella, in modo da permettere all'elemento di risaltare ed essere ben visibile anche all'interno di pagine web particolarmente colorate.

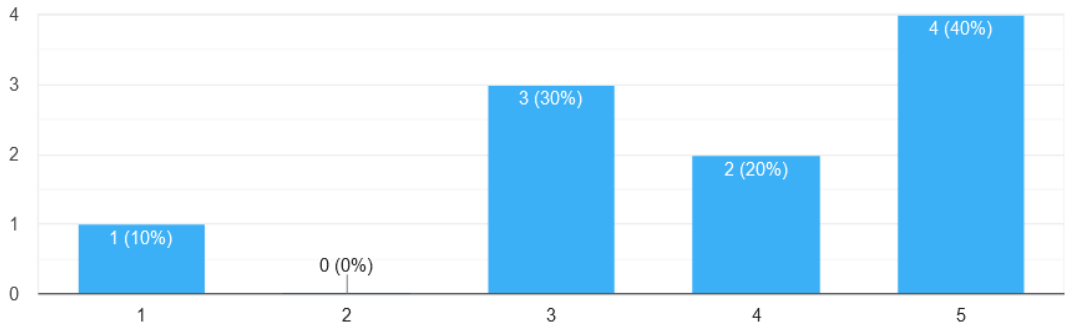

Figura 4.10. Valutazione dell'efficacia della stella nell'incoraggiare esplorazioni più approfondite.

4.4.3 Punteggio

Il punteggio, che costituisce il più importante elemento di Gamification introdotto nel tool in quanto è il principale mezzo con cui si cerca di fornire al tester un feedback riguardo alla propria performance, è una caratteristica comune nell'ambito del game design e praticamente tutti i partecipanti (eccezion fatta per un solo tester) conoscevano questo elemento e hanno ritenuto che sia stato implementato in una maniera a loro familiare.

La presenza del punteggio è stata generalmente apprezzata dalla maggior parte dei tester. Infatti, come si evince dal grafico in Figura 4.11, in cui sono rappresentate le risposte alla domanda 2.4 del questionario, più della metà dei tester ha fornito una valutazione positiva o molto positiva, mentre solo tre di loro hanno espresso un giudizio neutrale a riguardo.

Dal grafico presente in Figura 4.12, che invece rappresenta le risposte alla domanda 2.10 del questionario, risulta evidente come sia stata generalmente apprezzata anche l'organizzazione dei dati e delle informazioni relative alla sessione appena conclusa all'interno della schermata di riepilogo finale. Queste informazioni sono strettamente legate al punteggio poiché sono i dati con cui vengono calcolate le varie componenti che formano lo score finale. Nonostante il giudizio generalmente positivo, un tester ha espresso il desiderio di poter visualizzare il punteggio parziale man mano che si procede nell'esecuzione della propria sessione di testing, mentre un altro ritiene opportuno che venga visualizzato il valore di tutte le singole componenti all'interno della schermata di riepilogo, in modo da permettere di individuare

più facilmente gli aspetti più lacunosi della performance.

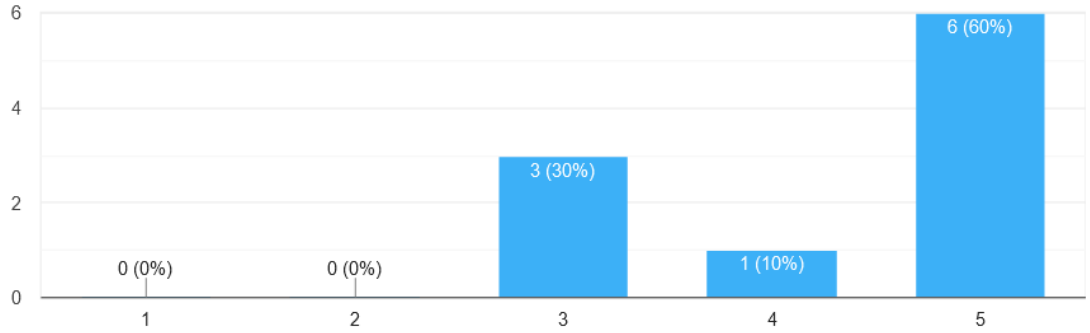

Figura 4.11. Valutazione dell'impatto motivazionale della presenza di un punteggio finale.

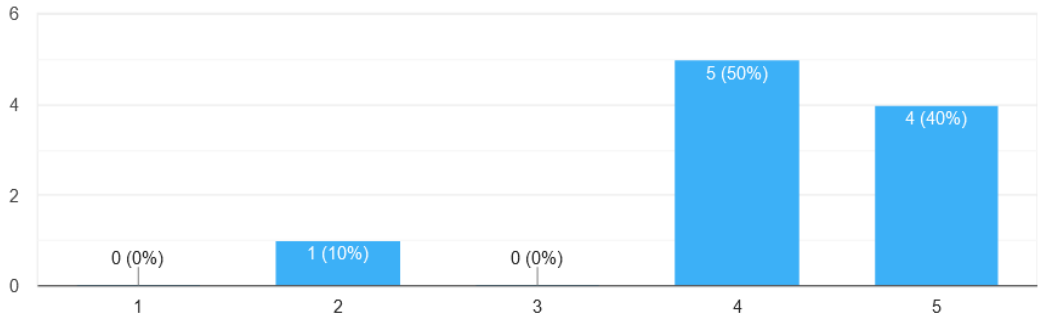

Figura 4.12. Valutazione della comprensibilità delle informazioni contenute nella schermata di riepilogo.

4.4.4 Classifica

Un altro elemento ricorrente nell'ambito del game design e della Gamification è la classifica degli utilizzatori dell'applicazione. Come risulta dalle risposte alle domande 2.7 e 2.8 del questionario, infatti, tutti i partecipanti erano a conoscenza dell'esistenza di questo elemento e hanno ritenuto che esso sia stato implementato in una maniera a loro familiare.

Nel caso particolare del nostro esperimento, come spiegato all'inizio di questo Capitolo, ogni tester si è trovato di fronte ad una classifica iniziale non vuota, visualizzabile solo una volta terminata la sessione di testing ludicizzata. L'efficacia

e l'impatto avuti dalla classifica sullo spirito competitivo dei tester possono essere valutati esaminando il grafico in Figura 4.13, che rappresenta le risposte alla domanda 2.3 del questionario: la quasi totalità dei tester ha dichiarato che la classifica abbia stimolato il loro senso di sfida nei confronti degli altri, eccezion fatta per un singolo partecipante che ha espresso un giudizio neutrale.

Infine, tra le considerazioni finali richieste tramite la domanda 2.15, un tester ha suggerito di separare la classifica attuale (che mostra i migliori dieci tester per punteggio complessivo ottenuto nel corso di una o più sessioni effettuate sul medesimo dominio) in una classifica che rispecchi quella attualmente esistente ed un'altra che invece mostri soltanto i punteggi relativi alle migliori performance singole eseguite su un particolare dominio.

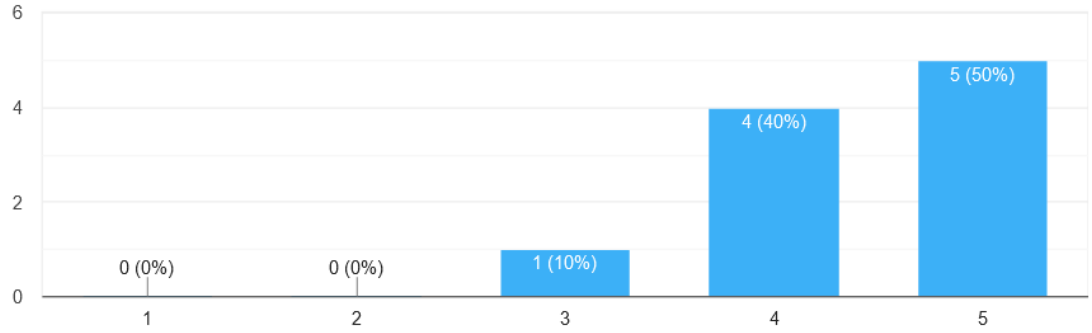

Figura 4.13. Valutazione dell'efficacia della classifica nello stimolare il senso di sfida dei tester.

4.4.5 Easter egg

L'ultimo elemento di Gamification inserito nel tool consiste negli easter egg, il cui scopo è quello di invogliare i tester ad esplorare più approfonditamente le pagine visitate in modo da scovare tutti quelli presenti. Anche questo è un elemento comune in molti videogiochi e rappresenta da sempre uno degli elementi più presenti all'interno di contesti non ludici: infatti, otto tester su dieci hanno affermato di conoscerlo, ma solo sei di loro hanno ritrovato l'utilizzo che si aspettavano nell'implementazione degli easter egg all'interno di Scout.

Stando a quanto è possibile dedurre dal grafico rappresentato in Figura 4.14, che raffigura le risposte raccolte relativamente alla domanda 2.6 del questionario, l'efficacia degli easter egg ha colpito positivamente solo la metà dei partecipanti, che

si sono mostrati generalmente più moderati nei propri giudizi riguardanti questo particolare elemento.

Parte delle cause dietro a questo risultato può essere cercata nell'aspetto grafico piuttosto semplice degli easter egg, caratteristica che è stata evidenziata negativamente da due tester nello spazio riservato ai suggerimenti (domanda 2.15).

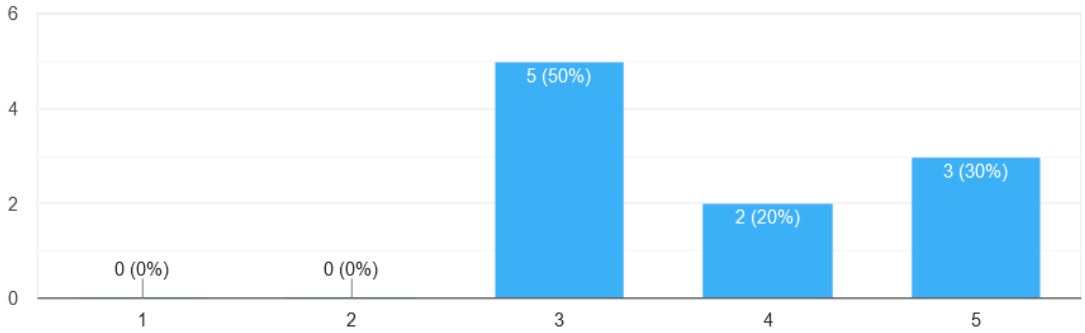

Figura 4.14. Valutazione dell'efficacia degli easter egg nell'invogliare i tester ad interagire con il maggior numero di widget all'interno di ogni pagina.

4.4.6 Giudizio complessivo

Nella parte conclusiva del questionario, è stata sottoposta ai tester una serie di domande per permettere loro di esprimere un giudizio complessivo sulla loro esperienza con il tool ludicizzato.

Prima di tutto, è stato chiesto ai partecipanti di selezionare quali elementi ludici abbiano avuto un impatto tale da rendere rilevante il loro eventuale inserimento all'interno della versione finale del tool (domanda 2.9). Le risposte ottenute sono visibili nel grafico rappresentato in Figura 4.15. Come si può notare, l'elemento che ha riscosso il maggior successo è stata la barra di progresso, apprezzata dalla totalità dei partecipanti. Classifica, punteggio ed easter egg hanno invece convinto appieno rispettivamente solo sei, cinque e quattro tester, confermando sostanzialmente quanto evidenziato dalle risposte alle domande specifiche poste in precedenza e spaccando in due i giudizi. La stella di evidenziazione delle pagine nuove, invece, non ha convinto nessuno dei tester ed è stato quindi ritenuto all'unanimità un elemento superfluo.

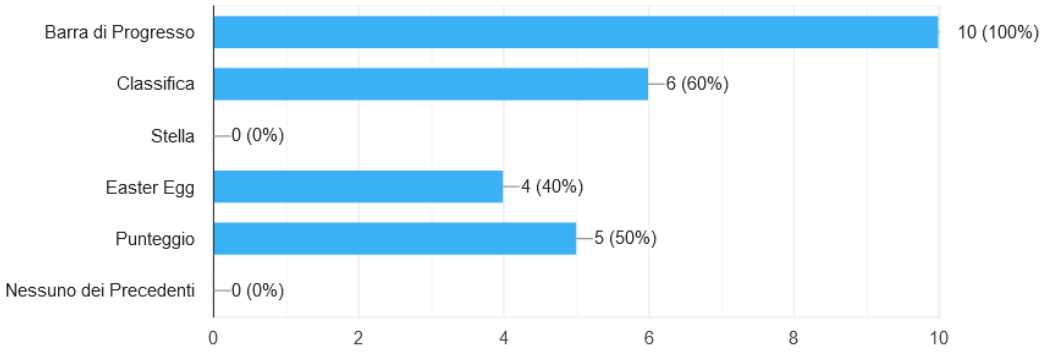

Figura 4.15. Numero di partecipanti convinti dalla presenza di uno specifico elemento ludico.

Successivamente, sono state poste delle domande riguardanti il tool nel suo complesso. Per prima cosa, nella domanda 2.11, è stato richiesto ai tester se ritenessero che il tool ludicizzato (Scout + plugin) possa essere integrato facilmente nel contesto di un ambiente di testing già esistente (lavorativo o di studio). Le risposte (Figura 4.16) sono moderatamente positive, con un solo tester che ha espresso un giudizio negativo.

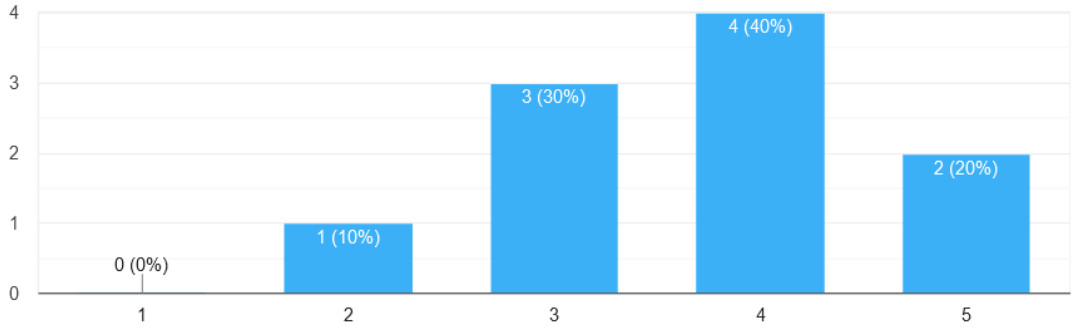

Figura 4.16. Valutazione della facilità di integrare il tool in un contesto di testing esistente.

In seguito, è stato richiesto ai tester di esprimere un giudizio sull'efficacia del plugin nel fornire un feedback riguardo i propri progressi e la propria performance nella sua totalità (domanda 2.12). Stando a quanto riportato dal grafico in Figura

4.17, tutti i tester hanno valutato positivamente questo aspetto, con sette di loro che hanno espresso un parere molto positivo.

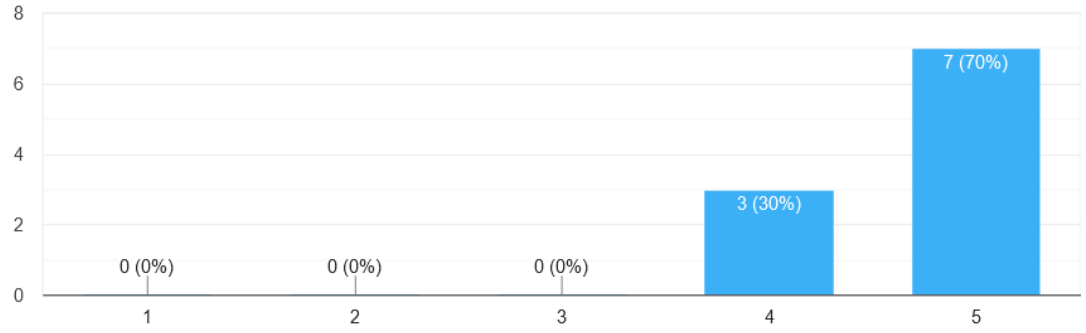

Figura 4.17. Valutazione della capacità degli elementi ludici di rendere i tester consapevoli dei propri progressi e dei propri risultati.

Infine, è stato richiesto ai tester di esprimere la propria preferenza tra la versione di Scout originale e quella ludicizzata (domanda 2.13). Il riscontro ottenuto, osservabile in Figura 4.18, volge fortemente a favore della versione gamificata del tool: sette partecipanti sono "decisamente d'accordo" sul fatto che la versione ludicizzata di Scout sia preferibile a quella originale, mentre i restanti tre sono semplicemente "d'accordo".

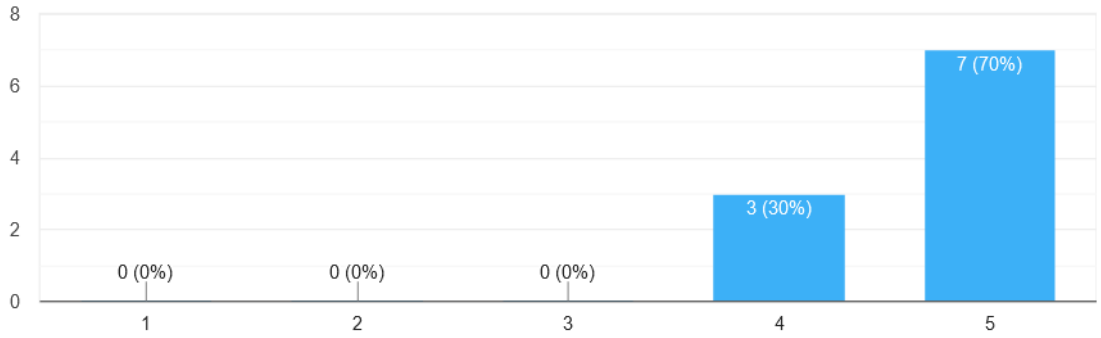

Figura 4.18. Valutazione sulla preferibilità della versione ludicizzata di Scout rispetto a quella originale.

Le risposte ricevute alla domanda aperta 2.14, volta alla segnalazione di eventuali problematiche riscontrate, hanno evidenziato criticità legate al funzionamento

di Scout (lag, perdita dello stato, difficoltà nell'utilizzo di alcune feature del tool originale), ma nessuno dei tester ha sottolineato problematiche vere e proprie relative agli elementi ludici inseriti.

Capitolo 5

Limiti di applicabilità

Il plugin realizzato per questa tesi aveva l'obiettivo di sperimentare l'efficacia dell'applicazione di alcuni elementi di Gamification al GUI testing di applicazioni web, ma la sua implementazione all'interno del contesto specifico di Scout ha costretto a dover fare delle assunzioni che hanno portato ad un campo di applicabilità più ristretto.

5.0.1 Limiti di validità interna

La validità interna si riferisce al modo in cui è stato condotto lo studio e riguarda eventuali variabili non considerate che possono aver portato i risultati osservati a non corrispondere in maniera causale al fenomeno studiato. Alcuni limiti sono legati a scelte implementative prese consapevolmente in fase di sviluppo, mentre altre sono legate a problematiche proprie di Scout.

Una delle principali scelte implementative riguarda la persistenza delle informazioni: si è scelto infatti di non utilizzare un database ma di rendere persistenti le informazioni riguardanti i tester, salvandole all'interno di un file testuale opportunamente scritto e parsificato a seconda delle necessità. La scelta di non adottare un database ha nettamente semplificato la maggior parte delle operazioni di accesso e salvataggio dei dati, ma allo stesso tempo ha ristretto il campo di utilizzo del tool, portando forzatamente a non poter effettuare una sincronizzazione delle sessioni in tempo reale, consentendo quindi di svolgere soltanto sessioni in tempi differenti, poiché non è possibile gestire le metriche che dipendono dall'istante di tempo in cui avviene l'interazione (come scoperta di pagine e widget nuovi). Tutto ciò ha portato a definire un meccanismo di assegnazione dei punti per la scoperta

di pagine e widget che va a favorire intrinsecamente il primo tester che effettua una sessione in un particolare dominio, in caso di mancato riazzeramento dello stato di partenza fra una sessione e la successiva (a differenza di quanto fatto nel corso dell'esperimento).

Un'altra scelta impelementativa riguarda invece l'inserimento dell'ID da parte del tester all'inizio di una sessione: infatti, sebbene Scout renda facoltativa l'immissione di un nickname per poter effettuare una sessione di testing, l'intero plugin si basa sulla possibilità di poter gestire i dati relativi a ciascun utente, il che rende fondamentale avere a disposizione degli ID univoci per poter identificare ciascun utente.

La maggior parte dei restanti limiti di applicabilità interna riguarda invece alcune criticità proprie di Scout, che hanno portato a dover compiere delle scelte implementative specifiche volte a far fronte a queste problematiche.

La criticità più evidente riguarda il modo in cui Scout mostra a schermo la pagina da testare: essa, infatti, risulta parzialmente zoomata rispetto alla sua versione originale. In più, Scout non permette di scrollare fino in fondo alcune pagine che non siano particolarmente corte: ciò rende impossibile il testing di tutti i widget presenti all'interno di alcuni siti, poiché risulta possibile interagire soltanto con i widget effettivamente visibili. Per far fronte a tale problematica, è stata adottata la soluzione di escludere dal conteggio dei widget totali, al momento del caricamento di una nuova pagina, tutti quei widget la cui posizione risultasse esterna alla posizione dell'area mostrata da Scout. Questa soluzione ha consentito ai tester di avere la possibilità di raggiungere il 100% di coverage anche all'interno di pagine non completamente testabili.

Oltre ai widget esclusi per ragioni di posizione, vengono tagliati fuori dal computo totale anche i widget aventi dimensione inferiore a 1x1 pixel. In alcuni casi, infatti, può succedere che Scout rilevi come interagibili degli elementi, generalmente molto piccoli, non visibili all'utente finale (quindi privi di significato), che renderebbero molto difficoltoso il tentativo di raggiungere la massima coverage possibile.

Oltre alle problematiche appena descritte, Scout presenta altre criticità che intaccano l'esperienza complessiva che si vive utilizzando il tool. Molti tester, infatti, hanno riscontrato ripetuti problemi di lag e responsività del tool, che hanno in generale rallentato l'esecuzione delle sessioni. Scout, inoltre, talvolta presenta dei

limiti relativi alla visualizzazione o all'individuazione del corretto stato delle pagine. Per quanto riguarda la visualizzazione, Scout presenta delle criticità quando si tratta di interagire con siti web che permettono di aprire dei pop-up in sovrimpressione senza che cambi la pagina corrente: in questi casi, infatti, tutti gli elementi grafici generati dall'Augmented Layer in caso di interazioni precedenti avvenute sulla pagina restano visibili in caso di comparsa di una finestra in sovrimpressione, andando quindi a creare delle situazioni piuttosto confusionarie.

Un'altra criticità riguarda la *type action* che Scout mette a disposizione dei tester per inserire del testo all'interno delle apposite caselle presenti in diversi siti web (barre di ricerca, form ecc.). L'inserimento di testo tramite quest'azione, infatti, fa avanzare lo stato di Scout, facendo sostanzialmente perdere i progressi (come gli highlighted widget) ottenuti fino a quel punto all'interno della pagina, nonostante essa di fatto non sia cambiata con il solo inserimento del testo. Per tale motivo, è stato necessario chiedere ai tester di non effettuare alcuna *type action* nel corso delle loro sessioni, per evitare di generare statistiche errate.

L'ultimo limite legato alla validità interna riguarda invece il calcolo del tempo totale di una sessione. Come descritto all'interno della Sezione 2.3, tale calcolo viene seguito con l'ausilio di un oggetto *Timing*, che fa scattare il conteggio alla pressione del tasto *Start* di Scout e lo termina alla pressione del tasto *Stop*. Si noti tuttavia che il valore ottenuto può differire dalla somma delle tempistiche di tutte le singole pagine, poiché il timer della pagina è interrotto prima dell'emulazione del click e riprende al caricamento nella pagina successiva. Ciò porta ad avere una discrepanza che in sessioni particolarmente durature con un alto tasso di esplorazione può anche essere dell'ordine di diversi secondi. Questa disparità si deve principalmente alla complessità delle operazioni di emulazione del click e di caricamento della pagina web da parte di Scout.

5.0.2 Limiti di validità esterna

Quando si parla di validità esterna si fa riferimento all'applicabilità dei risultati nel mondo reale, perciò all'adattabilità del tool sviluppato in un contesto diverso e generalmente più complesso di quello in cui è stato testato.

Il primo limite riguarda la possibilità di utilizzare efficacemente Scout per testare siti le cui pagine contengono una dose elevata di codice JavaScript: l'interazione con l'Augmented GUI, infatti, non supporta l'esecuzione di script come avviene

all'interno dei browser veri e propri. Tutto ciò rende le pagine contenenti del codice JavaScript difficili da testare nella loro interezza. Per far fronte a questa problematica, per la sperimentazione sono stati scelti due siti che contenessero principalmente elementi statici, in modo da poterne testare il più possibile. Oltre a ciò, il codice di Scout è stato modificato opportunamente in modo che ogni volta che si riconosce un click, oltre a registare l'interazione avvenuta, vengano effettuati dei controlli di validità dell'indirizzo selezionato: in particolare, si verifica che non siano presenti le sottostringhe "javascript:", "ftp:", "mailto:" e che il link non contenga il riferimento ad un fragment ("#").

Riguardo ai siti scelti per l'esperimento e citati in precedenza, è opportuno notare come essi presentino una complessità differente dal punto di vista del testing: mentre il sito di compravendita contiene un numero di pagine piuttosto elevato e sia in generale un sito abbastanza completo di informazioni e aree testuali (per quanto piuttosto statico), quello di e-shop risulta decisamente più semplice dal punto di vista degli elementi al suo interno, presentando un numero inferiore di pagine e, in generale, di widget interagibili. Tutto ciò può rendere meno complicato ottenere risultati migliori sul secondo e questo può in parte aver influenzato alcuni dei dati descritti all'interno della Sezione 4.3.1. È proprio per queste ragioni che si è scelto di adottare uno schema di valutazione di tipo AB-BA: l'idea è stata quella di testare i medesimi elementi di Gamification su siti di diversa complessità. Nonostante quanto detto finora, è importante notare come due tester che hanno svolto la sessione ludicizzata sul sito di compravendita siano comunque riusciti ad ottenere dei risultati simili a quelli ottenuti dai tester dell'altro gruppo, migliorando la propria coverage media.

Un'altra problematica legata ad aspetti che dipendono dal sito che si sceglie di testare riguarda, ancora una volta, gli URL delle pagine che appartengono al dominio scelto. Infatti, in molti siti web reali può capitare che una stessa pagina (a livello di contenuto) sia in realtà identificata da link diversi. Visitare in due moementi differenti la stessa pagina raggiungengola tramite due link diversi porta ad avere due stati associati alla stessa pagina, che verrà quindi rappresentata da due nodi distinti dell'albero degli stati della sessione. Ciò può risultare in una discrepanza tra il numero di pagine visitate che viene rappresentato nelle statistiche ed il numero di pagine uniche effettivamente visitate nel corso della sessione.

L'ultima criticità relativa al contesto di utilizzo riguarda il modo in cui i widget vengono identificati all'interno del plugin di Gamification: come descritto nella

Sezione 2.2, ai widget viene assegnato un identificativo che segue un preciso ordine di attribuzione. I primi elementi che si ricercano per capire come assegnare l'ID interno sono l'ID del tag HTML dell'elemento e, in mancanza di quest'ultimo, la classe del tag HTML. Molti siti, però, spesso sorvolano su queste due proprietà al momento della definizione di un elemento del DOM. Ciò porta ad avere più elementi, appartenenenti ad una stessa pagina, ad avere il medesimo ID all'interno dell'albero generato dal plugin di Gamification, creando delle collisioni che possono alterare il corretto conteggio dei widget e, di conseguenza, tutte le statistiche ad esso correlate.

Capitolo 6

Conclusioni

Nonostante il campione forzatamente ristretto con il quale è stata effettuata la sperimentazione, il lavoro svolto ha prodotto dei risultati interessanti dal punto di vista delle opinioni dei tester che hanno preso parte all'esperimento.

Dalle risposte ottenute in seguito alla somministrazione del questionario, emerge come alcuni elementi ludici possano avere delle potenzialità concrete nel contesto del GUI testing di applicazioni web. L'elemento che ha riscosso il maggior successo fra quelli implementati è stata sicuramente la barra di progresso, come si evince dai dati descritti nelle Sezioni 4.4.1 e 4.4.6. Questo risultato conferma quanto emerso da altri studi relativi alla Gamification: la presenza di un meccanismo di segnalazione dei progressi in tempo reale che permetta di percepire quanto, in un determinato momento, sia stato esaustivo l'operato del tester su una pagina, risulta essere di grande utilità per gli utilizzatori del tool. Ciò è probabilmente dovuto al fatto che la barra di progresso incarna perfettamente due elementi chiave che stanno alla base dell'idea di Gamification: essa fornisce infatti un feedback grafico riguardo ai progressi ottenuti fino ad un certo momento (aumentando quindi la consapevolezza del tester) e, in più, fissa un obiettivo concreto che l'utente può raggiungere, così da incentivarlo a fare tutto il possibile per realizzarlo.

Altri due elementi che hanno ricevuto pareri generalmente positivi sono stati il punteggio e la classifica, come si può evincere dall'analisi dei dati presentata nelle Sezioni 4.4.3, 4.4.4 e 4.4.6. Questi risultati confermano quanto già affermato in studi come (Platonova & Bērziša, 2017) e forniscono altre prove sull'efficacia di un altro concetto comunemente ritenuto fra i capisaldi della Gamification: fare leva sullo spirito competitivo degli utenti può spronarli ad avere performance migliori, perché

spinti dal desiderio di superare il punteggio dei propri colleghi. I commenti degli utenti che hanno preso parte alla sperimentazione suggeriscono le buone potenzialità dell'utilizzo di un sistema a punti anche nell'ambito del GUI testing, lasciando spazio ad ulteriori perferzionamenti in termini di calcolo effettivo del punteggio e rappresentazione a schermo delle informazioni ad esso relative.

Minor successo è stato invece riscosso dal meccanismo di identificazione delle pagine nuove e dalla caccia agli easter egg. Come si evince dai dati presentati nelle Sezioni 4.4.2, 4.4.5 e 4.4.6, la stella è stata valutata abbastanza positivamente da più della metà dei tester ma, nonostante ciò, nessuno di essi l'ha ritenuta un elemento ludico essenziale nel contesto di un ambiente di testing più completo. Gli easter egg, invece, hanno suscitato più perplessità nei tester, specialmente riguardo al loro aspetto grafico, anche se quattro partecipanti ne hanno valutato positivamente l'utilità. Nel complesso, è lecito dire che si possano considerare tali elementi come un surplus che non deve necessariamente costituire lo scheletro del plugin, ma eventualmente arricchire l'esperienza dell'utente. È però opportuno notare come la ristrettezza dell'insieme delle pagine individuate nel corso di ciascuna sessione e la semplicità grafica degli easter egg possano aver influito negativamente sulla percezione di tali elementi, ritenuti superflui rispetto agli altri. Ciò lascia quindi dei margini di miglioramento per quanto riguarda tali meccaniche, che, se adeguatamente sviluppate, potrebbero incontrare un consenso più ampio.

In conclusione, stando anche a quanto emerge dalle risposte ricevute relativamente alla domanda 2.13 del questionario, si può affermare che la Gamification migliori l'esperienza complessiva di GUI testing e che abbia dunque buone potenzialità di essere applicata con successo anche in questo ambito.

I risultati ottenuti suggeriscono diverse possibilità di perfezionamento del lavoro svolto. Di seguito, viene elencata una serie di possibili miglioramenti su cui concentrare i lavori futuri:

- Svolgere un'ulteriore sperimentazione con un numero più nutrito di partecipanti, in modo da avere a disposizione più dati per effettuare analisi più approfondite.

- Memorizzazione dei dati tramite un database centralizzato, con conseguente possibilità di gestire la sincronizzazione tra diverse sessioni di testing effettuate in contemporanea sul medesimo dominio.

- Implementazione di un profilo personale dove ogni utente possa trovare tutte le informazioni relative alle sue sessioni passate. Questo aprirebbe la strada all'inserimento di altri elementi ludici quali i badge e gli achievement.
- Perfezionamento del calcolo del punteggio.
- Revisione dell'aspetto grafico e implementativo del meccanismo di evidenziazione della scoperta di nuove pagine.
- Miglioramento grafico generale degli elementi ludici introdotti dal plugin.

Riferimenti bibliografici

Anderson P., N. T., & R., M. (2015, 06). Facilitating programming success in data science courses through gamified scaffolding and learn2mine. In (p. 99-104). doi: 10.1145/2729094.2742597

Berkling, K., & Thomas, C. (2013). Gamification of a software engineering course and a detailed analysis of the factors that lead to it's failure. In *2013 international conference on interactive collaborative learning (icl)* (p. 525-530).

Borjesson, E., & Feldt, R. (2012). Automated system testing using visual gui testing tools: A comparative study in industry. In *2012 ieee fifth international conference on software testing, verification and validation* (p. 350-359). doi: 10.1109/ICST.2012.115

Buckley, I., & Clarke, P. (2018, 01). An approach to teaching software testing supported by two different online content delivery methods.. doi: 10.18687/LACCEI2018.1.1.377

Costa, I., & Oliveira, S. (2019). A systematic strategy to teaching of exploratory testing using gamification. In *Proceedings of the 14th international conference on evaluation of novel approaches to software engineering* (p. 307–314). SCITEPRESS - Science and Technology Publications, Lda. doi: 10.5220/0007711603070314

Dal Sasso, T., Mocci, A., Lanza, M., & Mastrodicasa, E. (2017). How to gamify software engineering. In *Saner 2017 - 24th ieee international conference on software analysis, evolution, and reengineering* (p. 261-271). (Cited By :15) doi: 10.1109/SANER.2017.7884627

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011, 09). From game design elements to gamefulness: Defining gamification. In (Vol. 11, p. 9-15). doi: 10.1145/2181037.2181040

Fowler, M. (1999). *Refactoring: Improving the design of existing code.* Addison-Wesley Longman Publishing Co., Inc.

Fraser, G., Gambi, A., Kreis, M., & Rojas, J. M. (2019). Gamifying a software testing course with code defenders. In *Proc. of the acm technical symposium on computer science education (sigcse).* ACM. (To appear)

Herranz, E., Colomo-Palacios, R., & Amescua-Seco, A. (2013). Towards a new approach to supporting top managers in spi organizational change management. *Procedia Technology*, *9*, 129 - 138. doi: https://doi.org/10.1016/j.protcy.2013.12.014

Herranz, E., Colomo-Palacios, R., de Amescua, A., & Yilmaz, M. (2014, 06). Gamification as a disruptive factor in software process improvement initiatives. *Journal of Universal Computer Science*, *20*, 885-906.

Mollick, E., & Rothbard, N. (2014, 05). Mandatory fun: Consent, gamification and the impact of games at work.
doi: 10.2139/ssrn.2277103

Nass, M., Alégroth, E., & Feldt, R. (2019). Augmented testing: Industry feedback to shape a new testing technology. In *2019 ieee international conference on software testing, verification and validation workshops (icstw)* (p. 176-183). doi: 10.1109/ICSTW.2019.00048

Nass, M., Alégroth, E., & Feldt, R. (2020). On the industrial applicability of augmented testing: An empirical study. In *2020 ieee international conference on software testing,verification and validation workshops (icstw)* (p. 364-371). doi: 10.1109/ICSTW50294.2020.00065

Passos E.B., N. P. C. E., Medeiros D.B. (2011). Turning real-world software development into a game. In *2011 brazilian symposium on games and digital entertainment* (pp. 260–269).

Pedreira, O., García, F., Brisaboa, N., & Piattini, M. (2015). Gamification in software engineering – a systematic mapping. *Information and Software Technology*, *57*, 157 - 168. doi: 10.1016/j.infsof.2014.08.007

Platonova, V., & Bērziša, S. (2017, 12). Gamification in software development projects. *Information Technology and Management Science*, *20*. doi: 10.1515/itms-2017-0010

Prasetya, W., Leek, C., Melkonian, O., ten Tusscher, J., van Bergen, J., Everink, J., ... van Zon, W. (2019). Having fun in learning formal specifications. In *2019 ieee/acm 41st international conference on software engineering: Software engineering education and training (icse-seet)* (p. 192-196). doi: 10.1109/ICSE-SEET.2019.00028

Rojas, J. M., & Fraser, G. (2016). Code defenders: A mutation testing game. In *Proc. of the 11th international workshop on mutation analysis* (pp. 162–167). IEEE. doi: 10.1109/ICSTW.2016.43.

Santos, H., Durelli, V., Souza, M., Figueiredo, E., Silva, L., & Durelli, R. (2019, 09). Cleangame: Gamifying the identification of code smells. In (p. 437-446). doi: 10.1145/3350768.3352490

Sheth, S., Bell, J., & Kaiser, G. (2011). Halo (highly addictive, socially optimized) software engineering. In (p. 29–32). Association for Computing Machinery. doi: 10.1145/1984674.1984685

Zsigmond, I., Bocicor, M., & Molnar, A.-J. (2020, 01). Gamification based learning environment for computer science students. In (p. 556-563). doi: 10.5220/0009579305560563