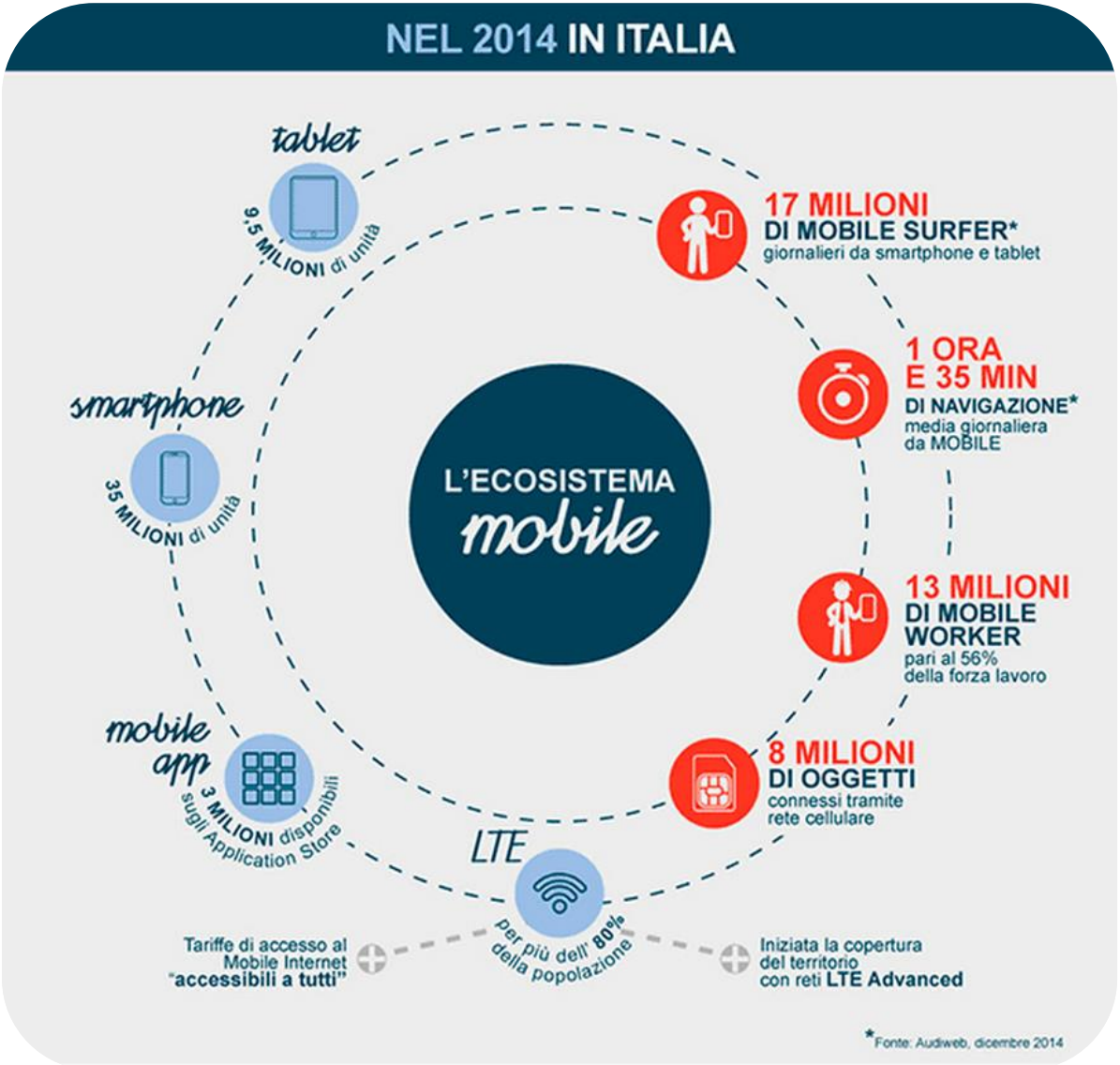
OSSERVATORI.NET
digital innovation

Strumenti e metodi di testing in cloud

OSSERVATORI.NET digital innovation

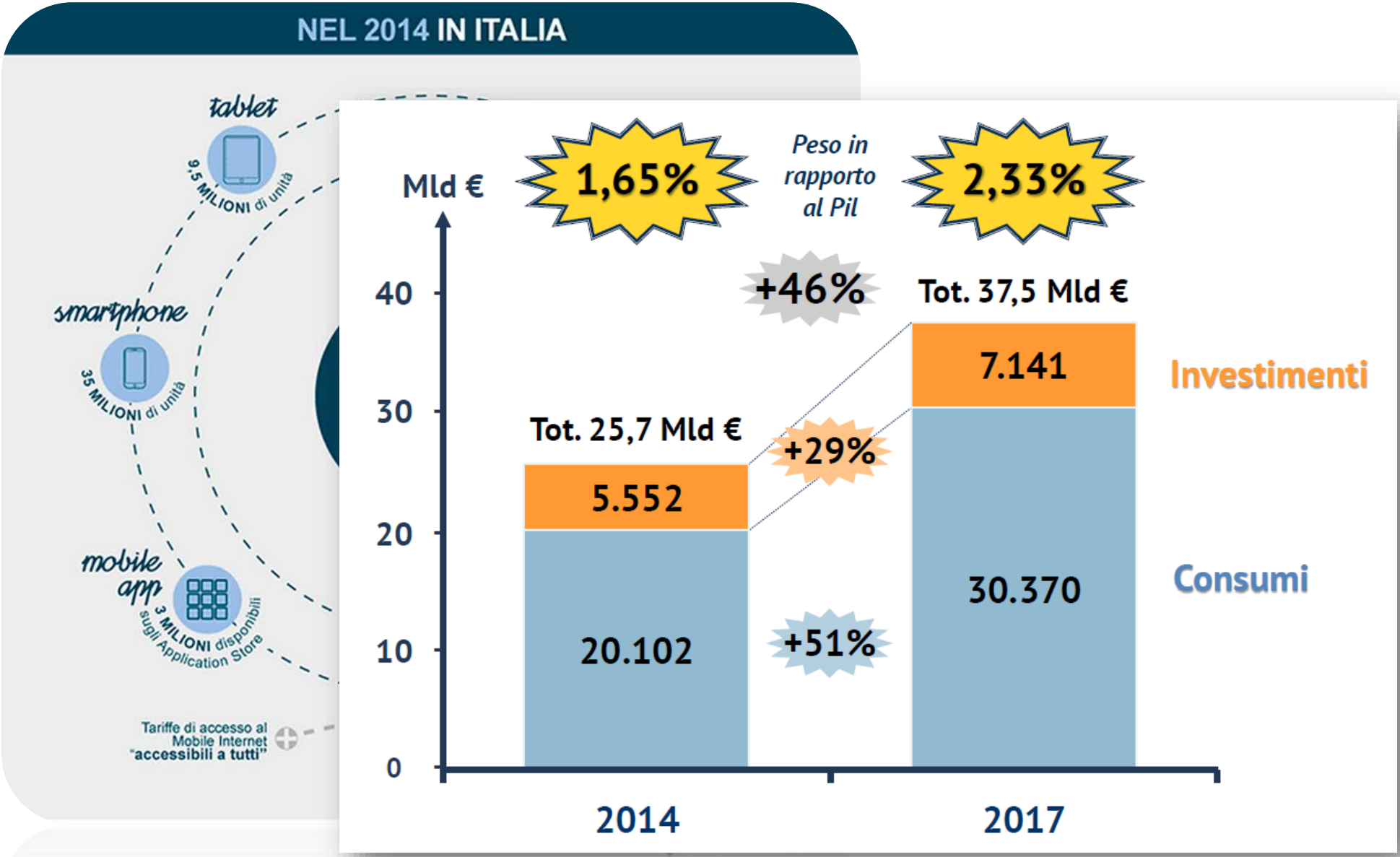
Perché parliamo di Testing Mobile



Source: www.osservatori.net

OSSERVATORI.NET digital innovation

Perché parliamo di Testing Mobile



Source: www.osservatori.net

OSSERVATORI.NET digital innovation

Perché parliamo di Testing Mobile

Rispetto al mondo Desktop e Web, lo sviluppo mobile ha delle caratteristiche che richiedono un cambiamento di paradigma nello sviluppo di servizi e prodotti per il mercato.

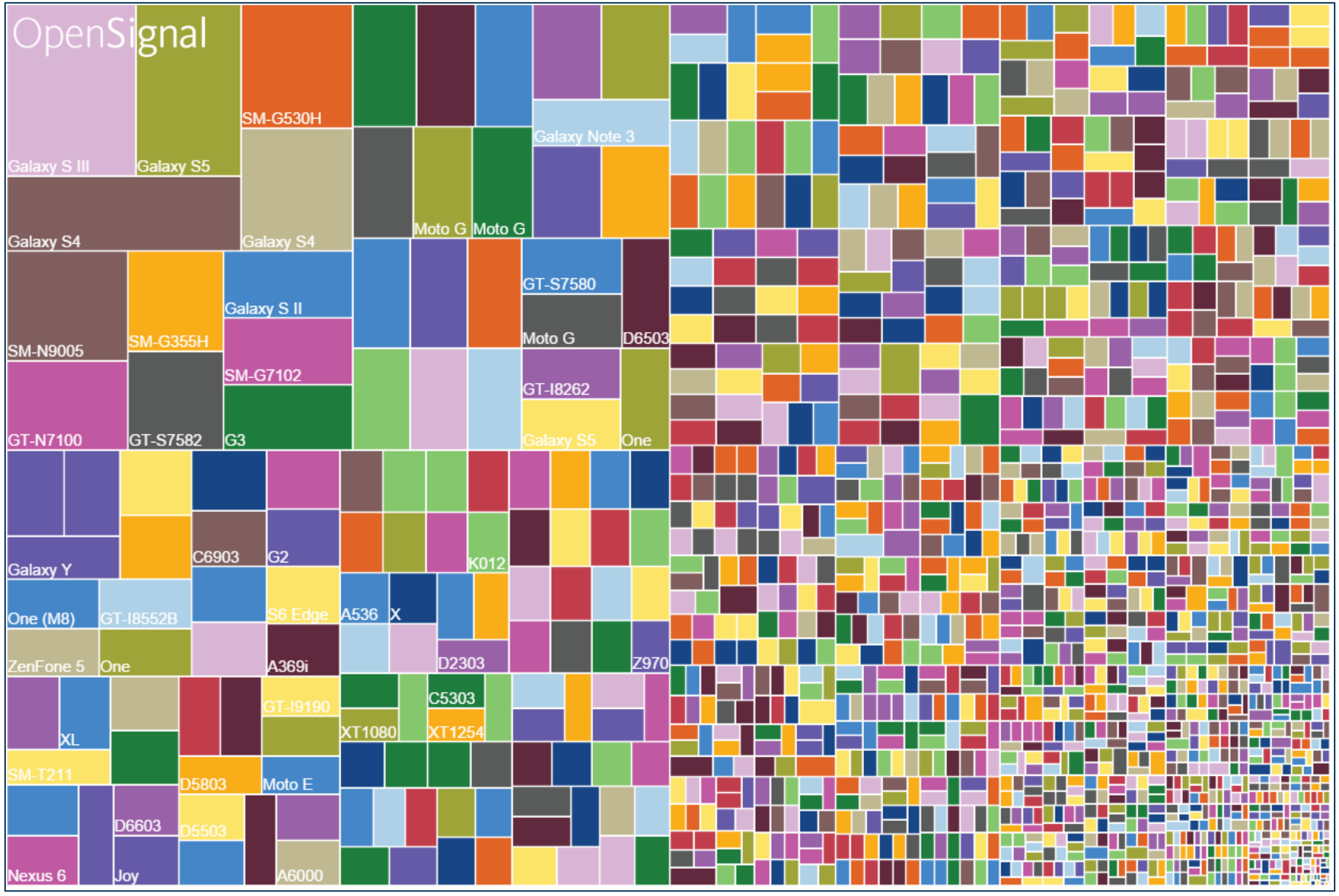
Le App non possono più semplicemente «funzionare»

Devono <u>DELIZIARE</u> gli utenti

Source: www.osservatori.net

OSSERVATORI.NET digital innovation

Esempio di complessità: Frammentazione device Android – Agosto 2015



Source: Open Signal - http://opensignal.com/reports/2015/08/android-fragmentation/

La complessità nel Mobile Development & Testing

Ecco alcuni aspetti da prendere sempre in considerazione nello sviluppo mobile che ne aumentano la complessità di gestione:

- Differenti piattaforme e device supportati
- Dimensioni dello schermo in continua evoluzione
- Complessità UI (User Interaction) → esiste sempre più di un modo di fare qualsiasi cosa
- Differenti tipi di applicazione – HTML5, Native or Hybrid
- Necessità di testare in ambiente reali (non su emulatori e simulatori)
- Problemi di sicurezza e privacy → molta attenzione sulla gestione dei dati
- Dipendenza da diversi operatori di rete, anche internazionali
- Installazioni, rimozioni e aggiornamenti rapidi → ambiente mutevole e in costante cambiamento
- Sessioni e interruzioni costanti → complessità esogena all'ambiente di sviluppo
- Test non-funzionali
- Risorse scarse: batteria, connessine dati, rete, GPS

OSSERVATORI.NET digital innovation

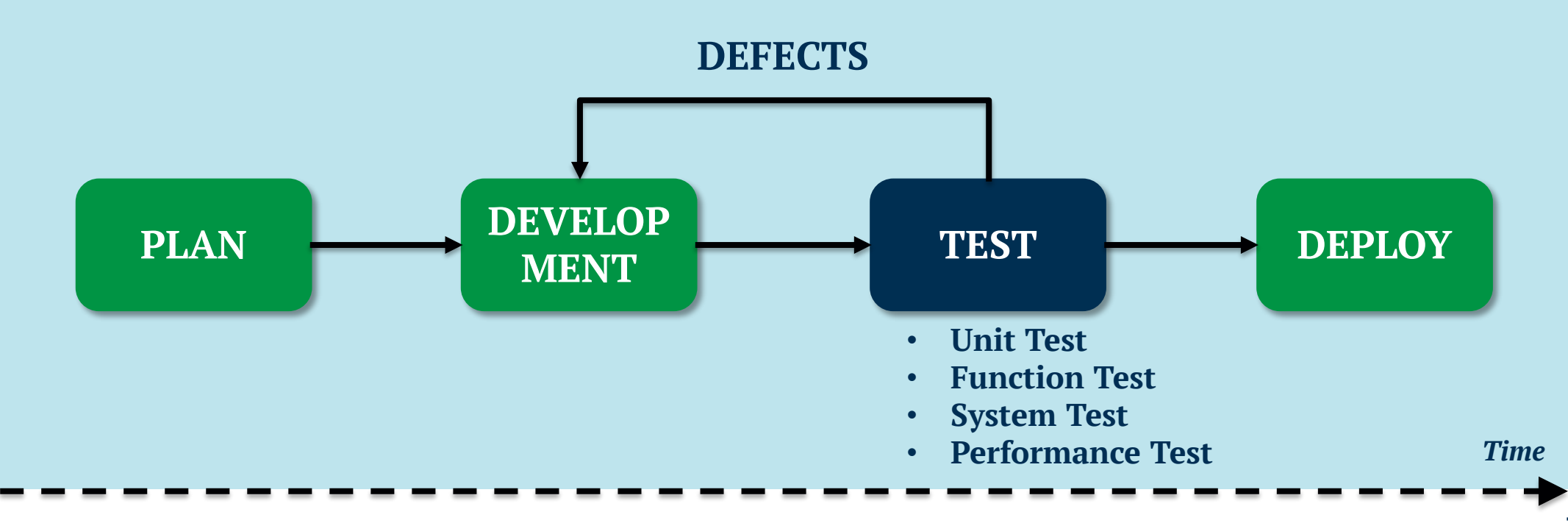
Processo di test: tutti coinvolti

OSSERVATORI.NET digital innovation

Il testing nello sviluppo tradizionale

Il testing nello sviluppo software tradizionale non si adatta ai nuovi paradigmi mobile, per diversi motivi:

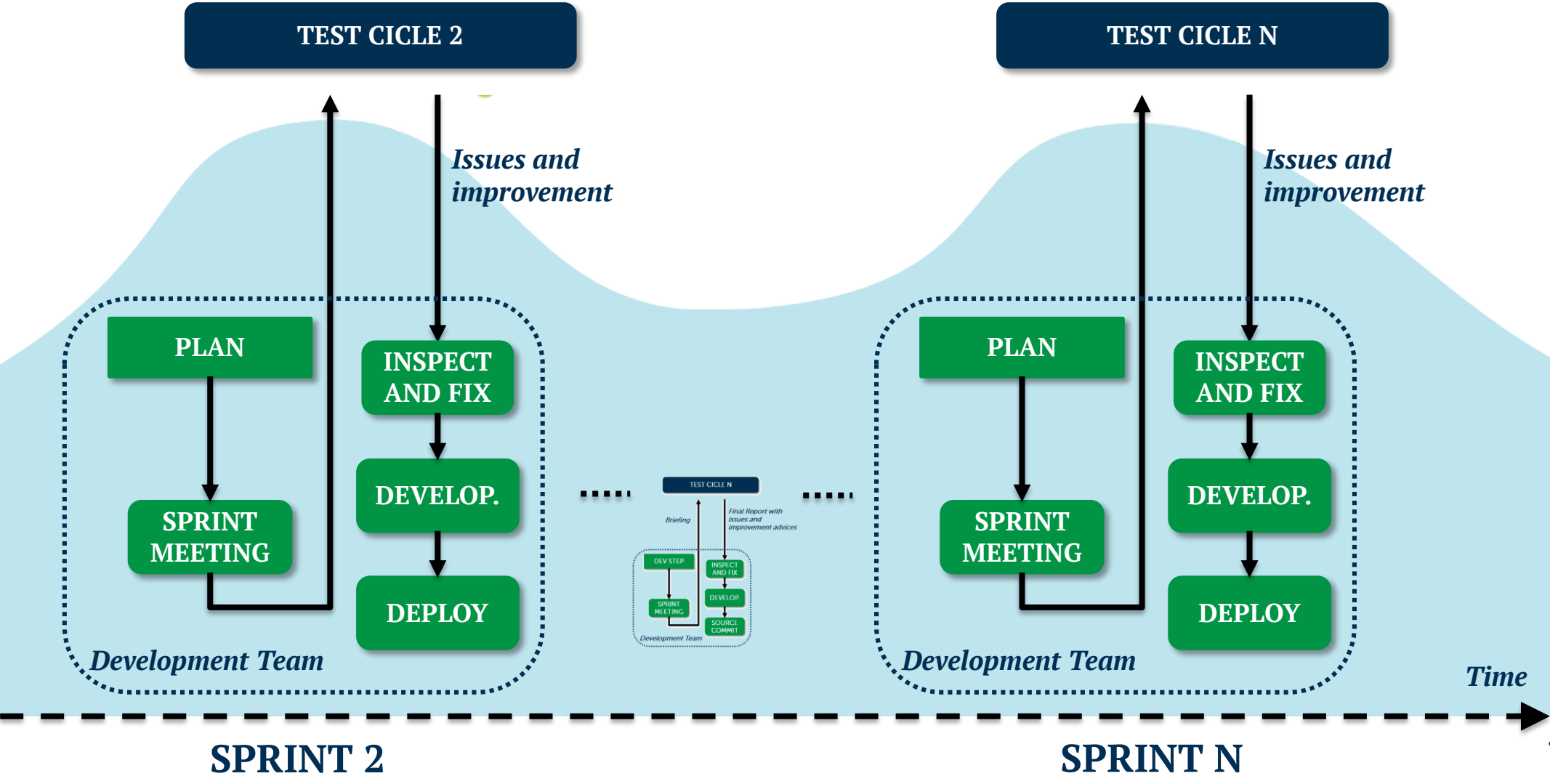
- Troppo lento e molto costoso rispetto alle esigenze di mercato
- Non si conosce cosa è sbagliato e cosa non funziona fino a che è troppo tardi
- Non considera il coinvolgimento del mondo e le condizioni «reali»
  - *While every bit of performance and uptime data is useful to understanding mobile application performance, in the end the only metric that matters is end-user satisfaction." — Aberdeen Group*



OSSERVATORI.NET digital innovation

Il testing nello sviluppo mobile

Il testing nello sviluppo mobile prevede l'utilizzo di tecniche, processi e strumenti che permettono di guidare lo sviluppo (*Test Driven Development*) allineandolo velocemente ai requisiti di business.



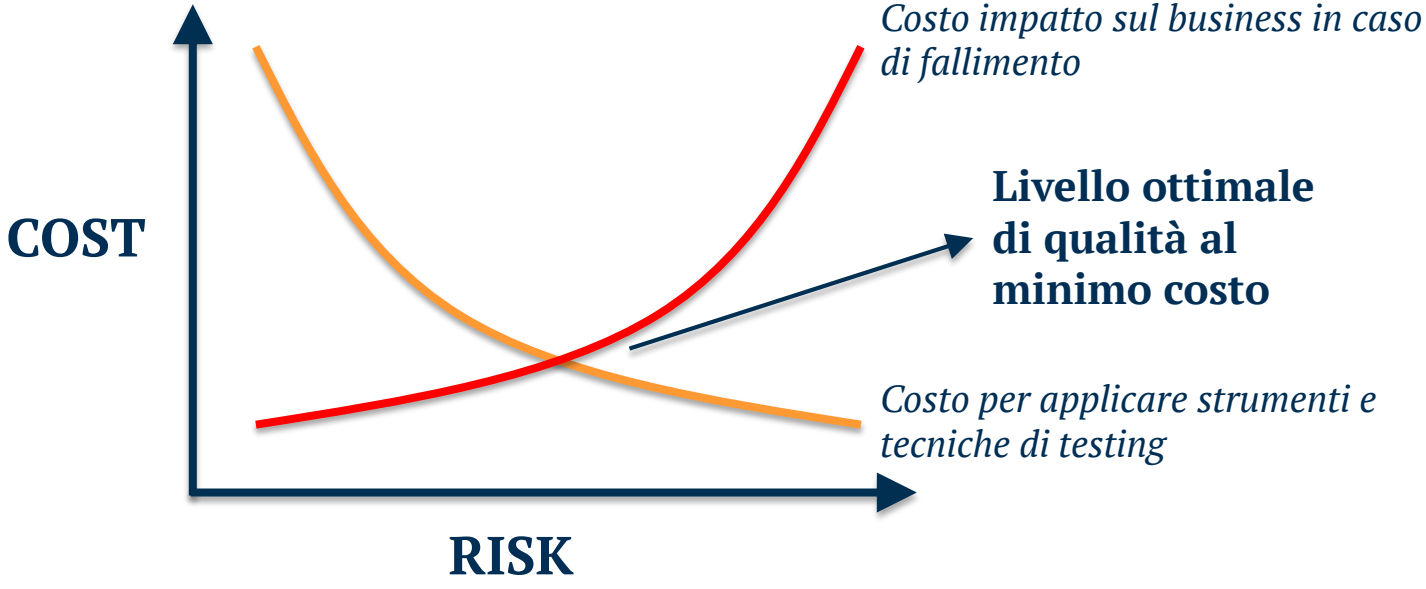
OSSERVATORI.NET digital innovation

Strategia di testing basata sul Risk Management

In un mercato in rapido cambiamento e isterico come quello dei servizi mobile diventa sempre più difficile gestire il rilascio di applicazioni di qualità e è necessario adottare un approccio basato sul **Risk Managment**.

Secondo questi metodi è fondamentale:

- **identificare i rischi** che incombono nello sviluppo e lancio dell'app;
- **classificarli** in base ai fattori probabilità di accadimento e impatto sul business;
- **prioritizzarli** in base a questi fattori;
- studiare azioni per **ottimizzare il costo.**

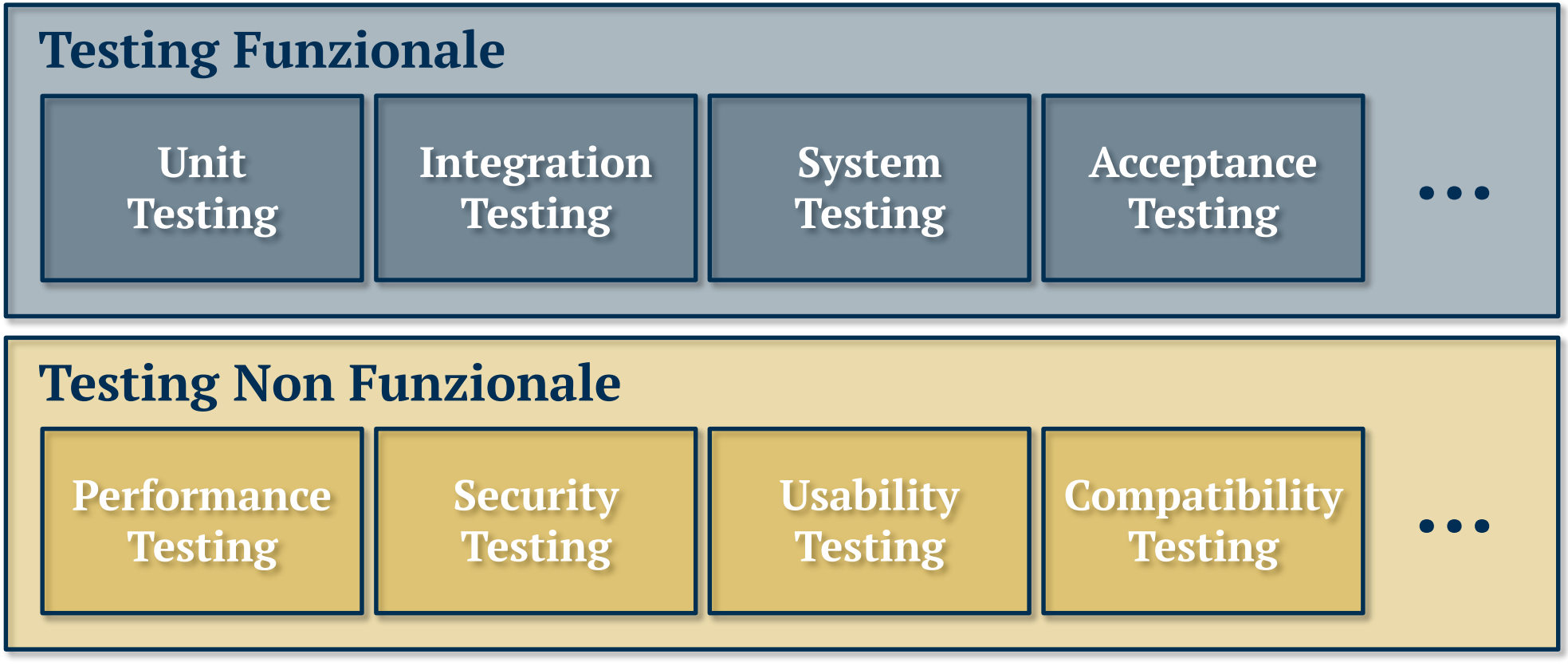


OSSERVATORI.NET digital innovation

Metodologie di testing

La classificazione principale in termini di Testing è suddivisa in 2 macro classi:

- **Testing funzionale**: test dell'applicazione nei confronti dei requisiti del progetto. L'obiettivo del testing funzionale è verificare che l'applicazione si comporti come è stata progettata: corrispondenza delle specifiche core.
- **Testing non funzionale**: test dell'applicazione nei confronti dei requisiti espressi (o non espressi!) non-funzionali. Tipicamente coinvolge misurare l'app vs parametri tecnici definiti (velocità, usabilità, vulnerabilità, ecc.)



Testing Funzionale

UNIT TESTING: Verifica la correttezza di ogni modulo o componente software. Scritto normalmente direttamente dagli sviluppatori.

INTEGRATION TESTING: verifica che diversi moduli o componenti unitari di software funzionino correttamente quando integrati assieme. Scritto normalmente direttamente dagli sviluppatori.

SYSTEM TESTING: implica la verifica che tutto il sistema sia esenti da problemi o bugs. Comporta il test sia Hardware che Software se previsto.

ACCEPTANCE TESTING: corrisponde alla fase finale del testing funzionale (chiamato anche UAT: User Acceptance Testing). Verifica che tutti i requisiti del progetto siano stati rispettati.

NON REGRESSION: verifica che una nuova feature dell'applicazione non va da a impattare sulle feature fino a quel momento funzionanti

Testing non funzionale

Il testing funzionale, quello normalmente svolto nello sviluppo di un'app, si basa sulla assunzione che le specifiche siano costatanti, precise e scritte correttamente a priori

E' necessario dotarsi quindi di strumenti di testing anche, e soprattutto, non funzionale!

Testing Non Funzionale

PERFORMANCE TESTING: ci possono essere diversi tipi di Performance Test ma hanno tutti l'obiettivo di ottimizzare il tempo di risposta dell'app che influiscono sulla fruibilità dell'app da parte dell'utente. Ad esempio possiamo misurare come l'app reagisce a picchi di utenti inaspettati o possiamo verificare i tempi di risposta del back-end (test di carico). Identifica anche la capacità del software di scalare secondo diversi parametri non funzionali (Test di Scalabilità)

SECURITY TESTING: verifica della robustezza dell'App o del Mobile Site agli standard di sicurezza odierni. Può riguardare la confidenzialità e integrità dei dati, falle nei sistemi di autenticazione, sistemi di non-ripudio, ecc.

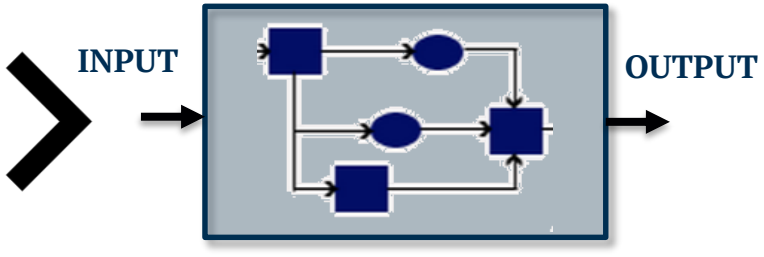
USABILITY TESTING: verifica che i principali elementi che consentono la specifica facilità d'uso dei servizi mobile siano adeguati (reattività dei processi, intuitività dell'interfaccia, accessibilità e fruibilità delle informazioni)

COMPATIBILITY TESTING: verifica del funzionamento dell'app su diversi device, SO, browser (e plugin) e dimensioni dello schermo

INTERNAZIONALIZZAZIONE & GEO: garantisce che il servizio mobile sia utilizzabile in differenti aree geografiche (anche internazionali)

Box Approach

WHITE BOX: questo approccio prevede la perfetta conoscenza della struttura interna del sistema . Sono necessari quindi forti competenze di programmazione e l'accesso a tutti i dati e componenti interni. Per esempio un testing di unità è un approccio tipicamente white box.



BLACK BOX: si esaminano le funzionalità del sistema senza avere nessuna conoscenza dell'implementazione interna. Si ha la sola conoscenza di quello che il sistema deve fare e non come viene esguito.

OSSERVATORI.NET digital innovation

Differenti tool di testing per differenti necessità

Tecnica	Metodologia di Testing	Vantaggi	Esempio Tool
Testing Automation	Funzionale (UAT, Non Regressione)	Strumenti di testing automation, aiuta a velocizzare i processi di testing e a testare l'app su molteplici devices	• SeleniumSE • Appium • Rational test Wordbench (IBM)
Testing Labs	Funzionale (UAT), Internazionalizzazione, Compatibilità	Permettono di testare l'applicazione utilizzando molti terminali reali	• Perfecto Mobile • TestDroid • AWS (former AppThwack) • Google Test Lab • Xamarin
Beta Testing	Funzionale (UAT), Usability	Strumento utile per gestire il team di Beta Testing per la verifica dell'app prima del rilascio	• TestFairy • TestFlightApp
A/B Testing	Usability	Strumenti di AB Testing utile per migliorare continuamente l'interfaccia grafica. Fornisce anche la possibilità di generare anche Heatmaps	• vwo.com • Optimizely • Apptimize

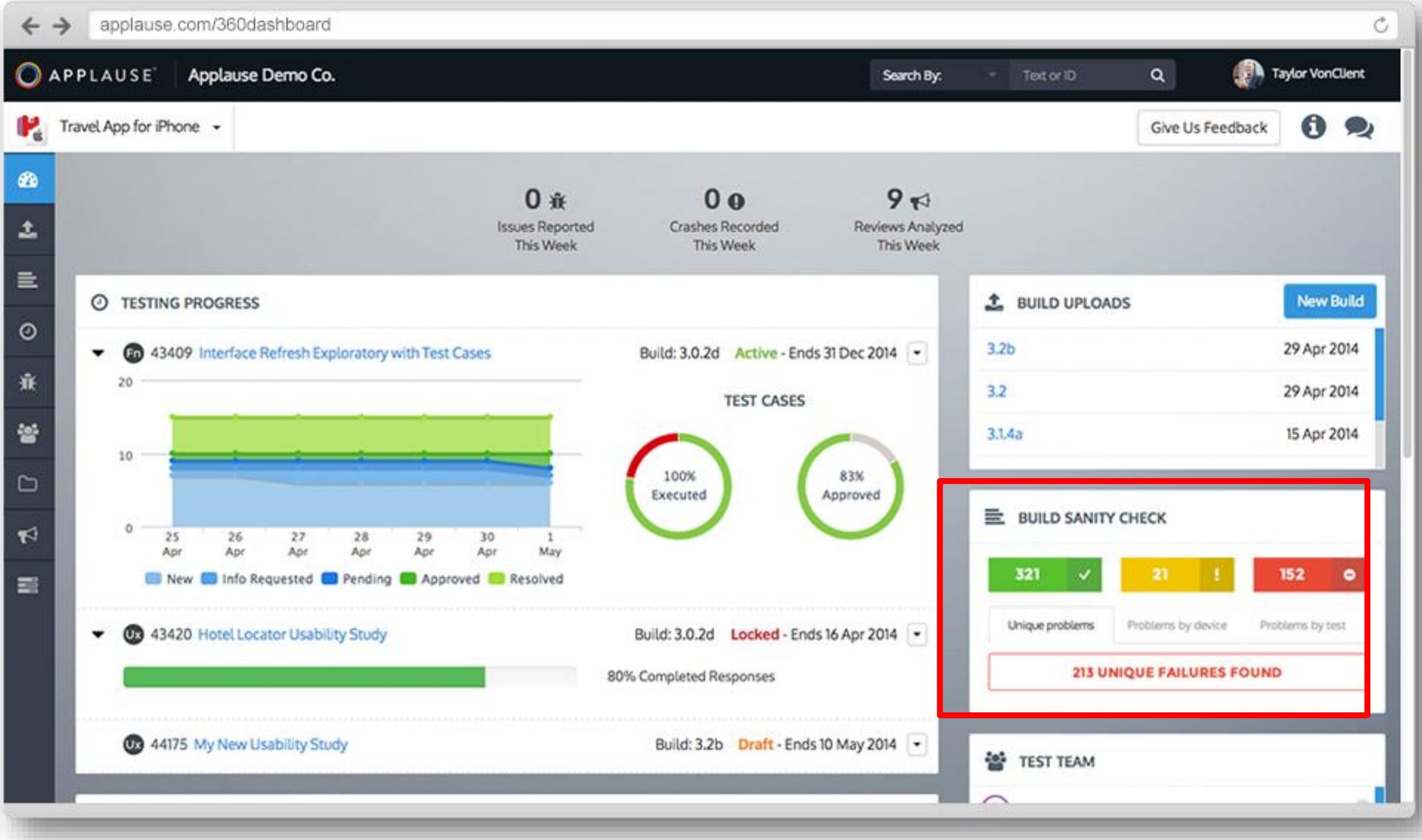
OSSERVATORI.NET digital innovation

Differenti tool di testing per differenti necessità

Tecnica	Metodologia di Testing	Vantaggi	Esempio Tool
Crowd Testing	Funzionale (UAT), Usabilità, Compatibilità	Permette di distribuire l'app a diverse persone e verificare il funzionamento in condizioni reali	• Applause • Testbirds
Performance Test	Carico, Performance, Scalabilità	Permette di verificare il numero di utenti virtuali che l'applicazione è in grado di supportare e lanciare centinaia di test in parallelo	• Blaze Meter • Neoload
Usability Test	Funzionale (UAT), Usability	Permettono di verificare e testare l'usabilità di un'app mobile (video degli utenti)	• UserTesting • UserZoom
Security Test	Sicurezza	Permette la verifica automatica di App mobile verificando le top10 OWASP	• AppScan (IBM) • Penetration Test

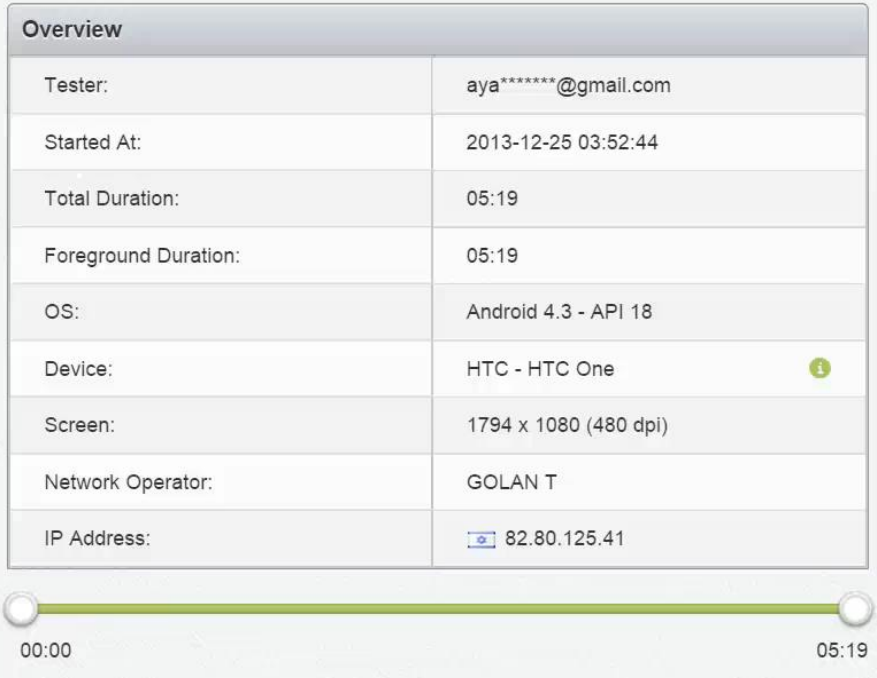
OSSERVATORI.NET digital innovation

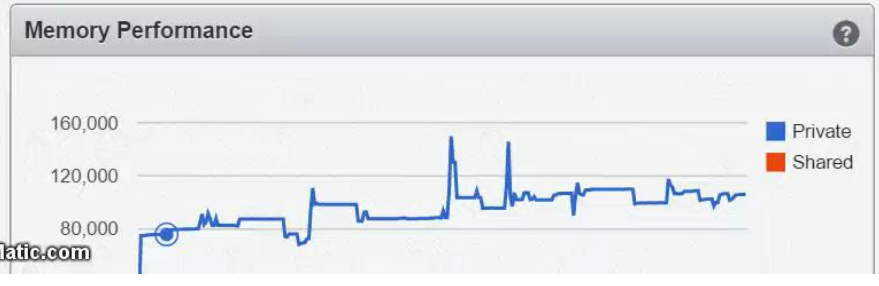
Esempio tecnica Crowd Testing Applause

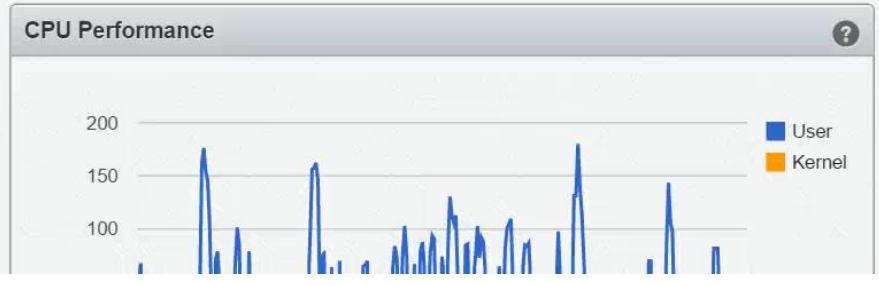


OSSERVATORI.NET digital innovation

Esempio tecnica Beta Testing TestFairy

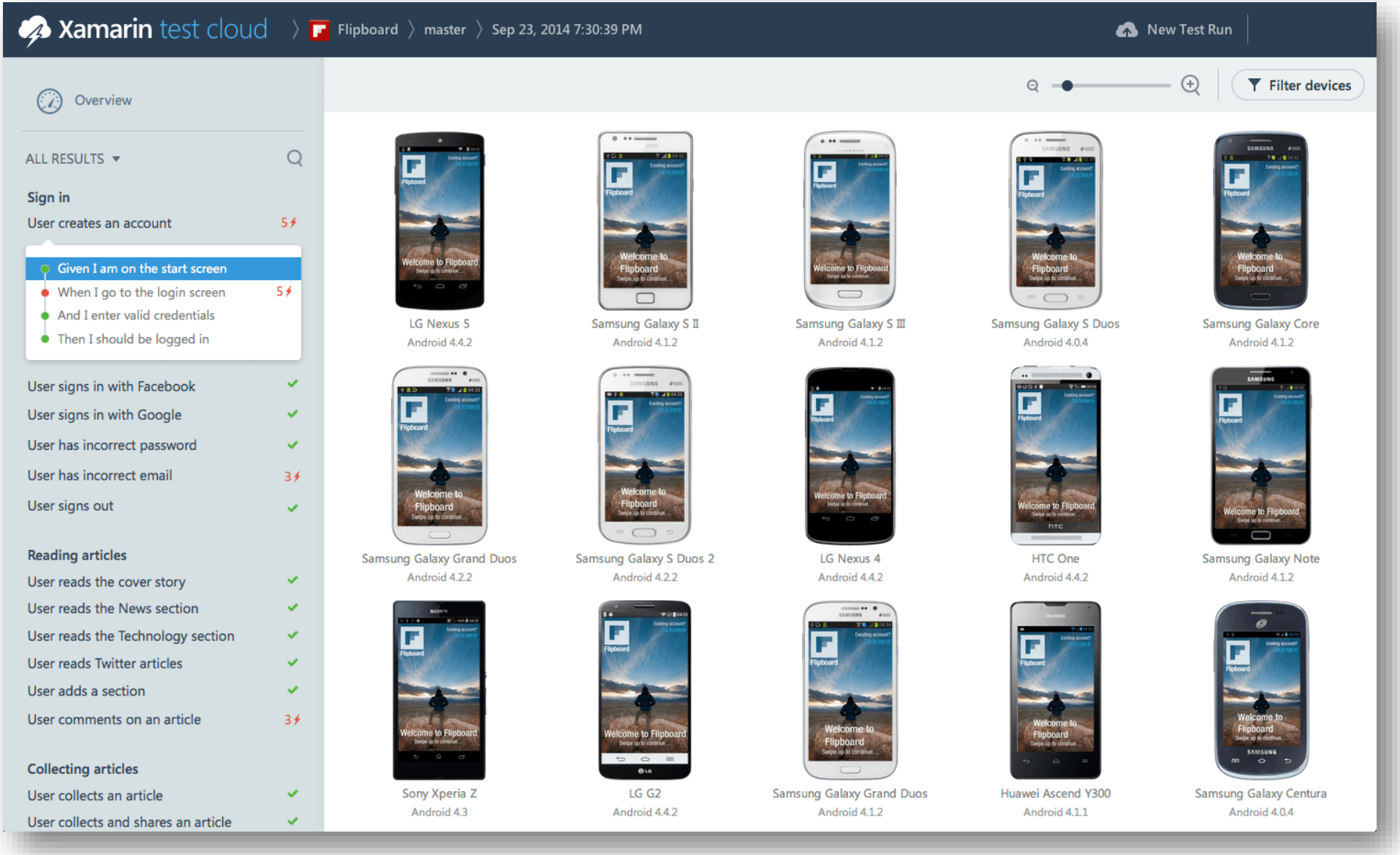






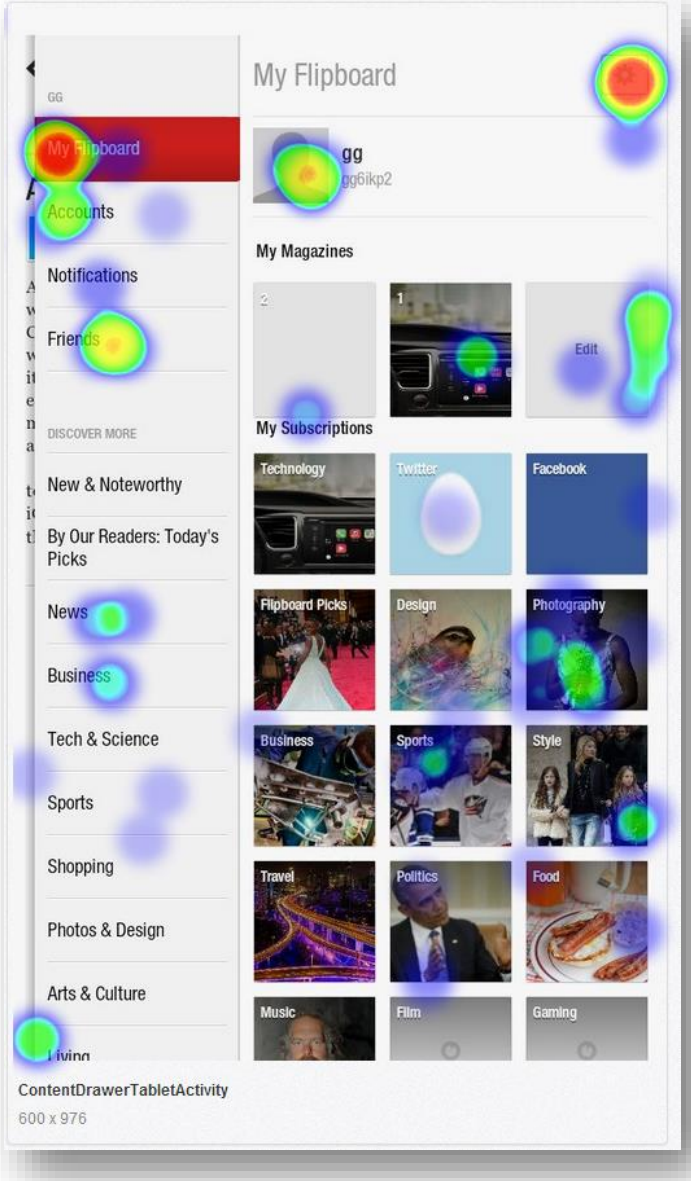
OSSERVATORI.NET digital innovation

Esempio Testing Labs Xamarin



OSSERVATORI.NET digital innovation

Esempio Usability Testing Heatmaps



OSSERVATORI.NET digital innovation

TCO: costruzione di una testing factory

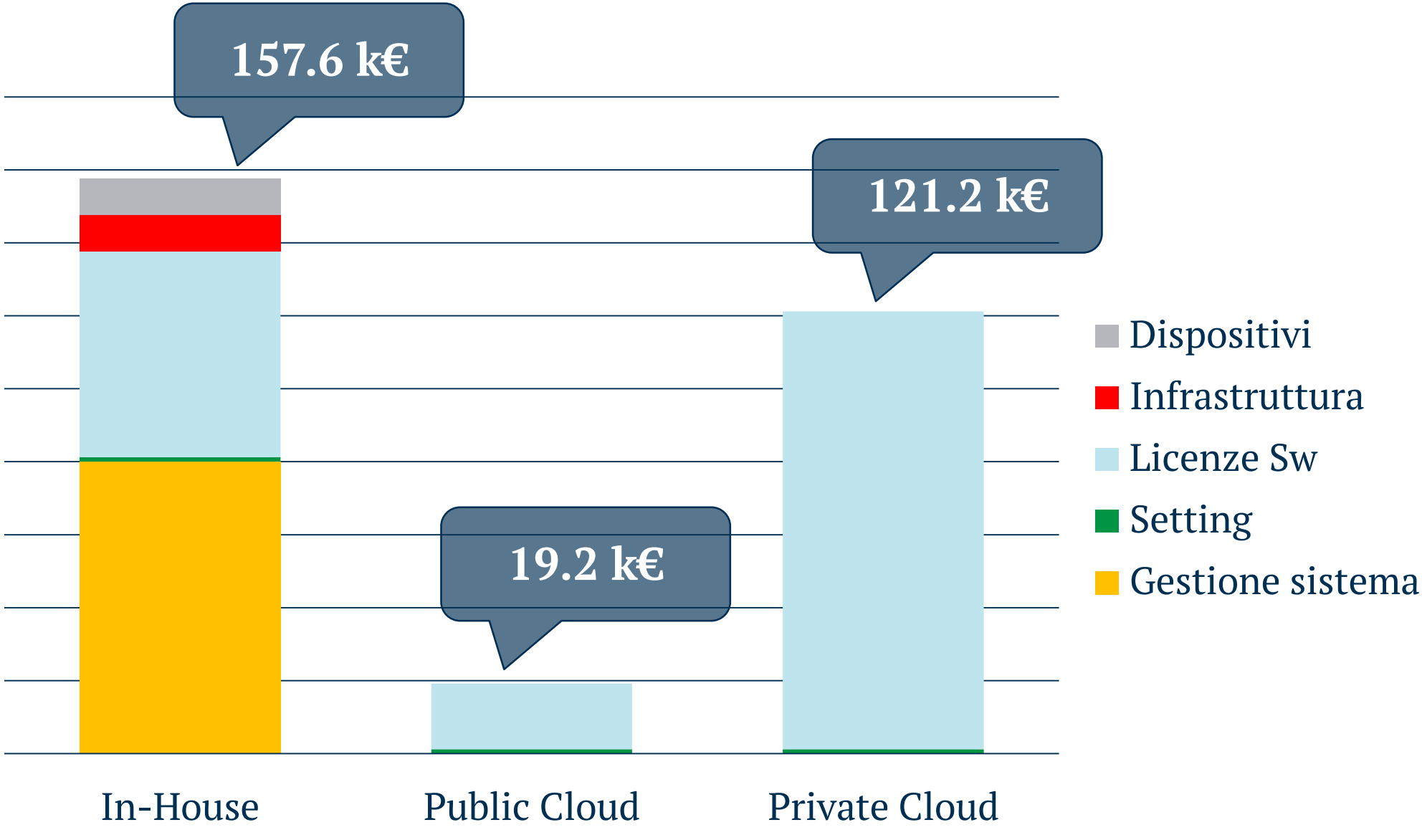
Esempio di calcolo TCO (Total Cost of Ownership): Costruzione di una testing factory internamente VS utilizzare sistemi as a service

ASSUNZIONI
– Sviluppo di diverse applicazioni
– Vengono utilizzati 50 device per il test sui dispositivi

Voci	In-house	Pubblic Cloud	Private Cloud
Dispositivi	200*50 = 10k€		
Infrastruttura Server	1000*10 devices = 10 k€		
Licenze Software	99/mese = 56.4 k€/anno	1.5k/mese = 18k€/anno	199/mese*50 = 120k€/anno
Setting Up & Deployment	3 gg lavorativi = 1.2 k€	3 gg lavorativi = 1.2k€	3 gg lavorativi = 1.2k€
Ore uomo gestione sistema (8/5)	1 FTE = 80k€/anno		

OSSERVATORI.NET digital innovation

Esempio TCO: costruzione di una testing factory



OSSERVATORI.NET digital innovation

Un team interno: oltre il TCO

Team Interno	Team Esterno
Copre efficacemente 2 device per persona e quindi spesso sono coperti non più di 6-8 devices	Con un lab di device esterno e tecniche di Testing Automation, 100 o più devices possono essere facilmente coperti allo stesso costo
Spesso svolge il testing come «in aggiunta» rispetto al proprio day-by-day e risulta difficilmente scalabile	Team scalabili significa che la copertura delle fase di testing è garantita e eseguita in tempi ridotti e impegnata solo quando necessario
Possono essere necessarie settimane per raggiungere la copertura necessaria a scenari multipli	Team dedicati possono portare risultati per cui task di testing possono essere eseguiti in pochi giorni
Il testing viene «in genere» eseguito dallo stesso team di sviluppo e, conseguentemente, problemi anche importanti sono identificati a fatica	«Fresh Eyes» e modelli «commission driven» significa che i problemi vengono identificati sistematicamente
Si focalizzano giustamente su (poche) competenze di sviluppo core con il relativo problema di aggiornamenti agli ultimi strumenti più evoluti	Usando fonti e esperienza esterna si può avere la possibilità di usare gli ultimi e più efficaci strumenti con il «giusto» investimento (tempo e soldi)

OSSERVATORI.NET digital innovation

6 Motivi per utilizzare differenti tecniche di testing non funzionali

Ridurre il time to market

1.

Few days test cycles (including weekends), no lag time, no issue with delayed development

Ridurre i tempi di sviluppo

2.

Cost savings are not only in QA, but also reduced development cost as bugs are found quickly and with a lower number of test cycles

Focus nello sviluppo delle features chiave

3.

You can focus on what really matters – build the perfect killer application – with our team assuring flawless apps and high quality standards

Frammentazione dei devices

4.

Our process can easily adapt your needs covering many devices and many technologies facing, in this way, the issue of fragmentation

Feedbacks dal mondo reale

5.

Using the biggest crowd in the world we are able to test on the right and real target of your business. This is a complete different approach on having only tech feedbacks

Certificazione esterna

6.

With the strong experience of our team and the obsession on finding problems you can trust our external point of view

Altri importanti motivi: Short kick off times & flexible price

OSSERVATORI.NET digital innovation

Case study: Crowd testing

	CASE STUDY DESCRIPTION
Quando	Giugno 2015, prima del “Lancio Ufficiale”
Cliente	Società di Utility Media Italiana
Test Concept	Bugs Finding + Functional Testing Trovare il Massimo numero di problemi e supportare il team di sviluppo nella veloce risoluzione degli stessi prima della pubblicazione dell’app sugli store
Test Details	Android + iOS Crowd Test 80 Devices **Lead Time: 2 weeks**
Results	**~2.500 Test Session** **~60 Anomalie Rilevate** **Usability Video + Heatmaps + Customer Screenshots**

OSSERVATORI.NET digital innovation

Case study: Competence Centre + Testing End to End Automation (Test Factory)

	CASE STUDY DESCRIPTION
Quando	2014 – Full year
Cliente	Banca internazinale olandese
Test Concept	Ridurre il processo manuale e gli errori con processi e standardizzati e automatici. Migliorare i processi di sviluppo Dev Ops → permettendo di rilasciare aggiornamenti e miglioramenti all'app più velocemente e molto più spesso
Test Details	Android + iOS Testing Automation process + Testing Labs Lead Time: 8months (change management)
Results	**Increase throughput from 4 weeks to 6 hours** **Avarage time to market from 13 weeks to 1 week** **Test covereage increase from 30% to 80%**

OSSERVATORI.NET digital innovation

Alcuni consigli

- Pianificate diverse release partendo da funzionalità ben specifiche ma che funzionano bene – **DevOps e Continuous Delivery**
- **Automatizzare** dove possibile (in ogni fase, anche in **UAT**!)
- Create **strutture flessibili**
- **Outsourcing** vs Insourcing non solo dello sviluppo, anche del testing
- Create un **centro di competenza** Sviluppo & Testing Mobile
- Usate anche tecniche di **testing non funzionale**
- Non dimenticate le **performance** (nel mobile sono critiche)
- **Approcci qualitativi** e non solo quantitativi
- Verificate il **mondo reale** in continuazione dalla prototipizzazione al delivery
- Gli **Analytics** sono un must

Golden Rule

First Law Of Software Quality

$$E = mc^2$$

$$\text{Errors} = (\text{more code})^2$$

Testing Automatizzato

❑ Cosa è il test automatizzato:

- Quali sono le aree di intervento
- Come si costruisce
- Quali frameworks e quali tools
- Limiti di applicabilità
- Principali differenze tra i frameworks

Testing Automatizzato - aree di intervento

- **Quali sono le aree di intervento:**
  - particolarmente efficace nel test funzionale
  - Identifica nel processo esecuzione dell'applicazioni:
    - bugs dovuti a cattiva programmazione
    - Incongruenze con le specifiche funzionali
  - I test possono essere eseguiti ad un costo ridotto e strettamente proporzionale al numero di dispositivi su cui viene eseguito
    - Consente di identificare in modo efficace la frammentazione dei terminali
    - Consente di dominare la congruenza tra le diverse versioni della stessa applicazione (test di non regression)

Testing Automatizzato - aree di intervento

❑ NOTE:

- Il processo di scrittura dei test può essere eseguita parallelamente alla fase di sviluppo (*approccio preferibile*) oppure in fase di UAT
- Il set di test creati per verificare l'applicazione diventa un importante assett del progetto, per verifiche future:
  - Nuovi dispositivi in commercio
  - Nuove versione del sistema operativo
  - Nuove release dell'applicazione
  - Nuove release del backend applicativo
- Monitorare progressivamente lo stato di maturità dell'applicazione in sviluppo

Testing Automatizzato – come costruire

Come si costruisce:

1. Identificare i percorsi funzionali presenti nell'app/msite in test
2. attraverso un opportuno framework applicativo vengono scritti dei casi di test: tali porzioni di codice consentono di simulare su **device reali,** o simulati, l'interazione che l'utente avrebbe con il device (smartphone o tablet)
3. Gli script di test vengono eseguiti su **device reali** o simulati presenti in locale od in **remoto**. Ogni script di test prevede la verifica di più elementi strutturali e funzionali in un unico percorso.
4. Al termine del processo viene raccolto un risultato positivo o negativo per ciascun test eseguito su ciascun device.

Testing Automatizzato - framework e quali tools

- **Quali framework e quali tools:**
  Sul mercato vi sono molte soluzioni per eseguire test in modo automatizzato, la quasi totalità si basa su analoghe soluzione elaborate per il mondo web **(WebDriver)**
  - I fattori decisioni per la selezione del framework opportuno dipende da differenti fattori
    - Il grado di interazione con il dispositivo
    - Le piattaforme [HTML5, iOS, Android] che si intendono indirizzare con il test e le relative versioni
    - Il linguaggio in con cui scrivere i test
    - Il framework supportato del dal provider dei servizi
    - La complessità del dei test che si devono realizzare

Testing Automatizzato - Limiti

❑ Limiti di applicabilità

- Non è possibile testare tutto: si può ottenere una ottima copertura ma è difficile ottenere una copertura completa.
- Un test di questo tipo non sostituisce ne il test d'unità che il test d'integrazione
- Non è bene testare l'integrazione ma solo la singola feature: Ei (nel processo di login: controllo che il login avvenga con credenziali corretti, controllo di avere un opportuno messaggio di errore se si logga con credenziali non corrette. Non faccio ulteriori verifiche sul backend)

Testing Automatizzato - Confronto tra frameWork

❑ Tabella per piattaforma

	Calabash	Robotium	Uiautomator	Espresso	Appium
Android	SI	SI	SI	SI	SI
iOS	SI	NO	NO	NO	SI
Mobile Web	SI* per applicazioni ibride	SI[1]	SI[2]	NO	SI

NOTE:

1. Solo sula piattaforma Android
2. Solo in termini di click su coordinate delle schermo (X & Y) ed ovviamente solo piattaforma Android

Testing Automatizzato - Confronto tra frameWork

❑ Tabella per framework

	Calabash	Robotium	Uiautomator	Espresso	Appium
Linguaggio	Ruby	Java	Java	Java	Java, js, c, …
Tool di creazione dei test	Tool a riga di comando	Registratore grafico	Strumento grafico per identificare gli elementi grafici con cui interagire	Strumento grafico per identificare gli elementi grafici con cui interagire	Registratore grafico
Versioni supportate	Supporto completo	Supporto completo	Android sdk > 16	Android sdk: 8,10, 15-23	Android sdk > 17

Testing Crowd - aree di intervento

- **Il Crowd test ha un range di applicabilità molto esteso:**
  - Identificazione di bugs funzionali
  - Identificazione di bugs o issues di usabilità
  - Identificazione di bugs dovuti a :
    - Utilizzo di un device nelle **condizioni** reali d'uso quotidiano: più app aperte, differenti app installate, …
    - Utilizzo di un device nel **contesto** reale d'uso quotidiano: diversi operatori, diverse condizioni di connettività [livello del segnale, latenza della rete, operatore, policy di sicurezza]

Testing Crowd - Costruzione del test

❑ per costruire un Crowd :

- identificazione dei percorsi da verificare
  - se **blind** test (viene lasciato libero accesso alla app): il crowd non è preistruito
  - se non **blind** test: predisporre documentazione per istruire il crowd in modo che sollecitino le sezioni e le funzionalità desiderate
- selezione del crowd secondo 3 vettori :
  - tipologia di terminale posseduto
  - parametri demografici [target dell'applicativo]
  - livello di confidenza pregressa con l'app/processo in esame
- identificazione della numerosità del crowd
  - mediamente da 10 a 30 soggetti
  - generalmente la numerosità del crowd è inversamente proporzionale alla complessità del task che viene sottoposto

Testing Crowd – Limiti di Applicabilità

- Limiti di applicabilità
  - Il Crowd è in grado di svolgere anche task complessi che coinvolgono elementi esterni (utilizzo in mobilità, accessori, account personali) ma la complessità del processo di istruzione del crowd è direttamente proporzionale alla complessità dei task sottoposti, così come la complessità della processo di raccolta dei dati.
  - la copertura dei device è generalmente minore rispetto ad un test di automazione
  - per quanto ben strutturato il risultato dei test deve essere successivamente processato manualmente o semi manualmente
  - per quanto bene preistruito il crowd necessita di un presidio forte

OSSERVATORI.NET
digital innovation

Grazie

per maggiori informazioni:

Luca Manara:
cell +39 328 635 0983
email luca.manara@app-quality.com

Edoardo Vannutelli:
cell +39 349 291 9298
email edoardo.vannutelli@app-quality.com