POLITECNICO DI TORINO

Corso di Laurea Magistrale
in
Ingegneria Gestionale

Tesi di Laurea Magistrale

La gestione del Processo di Sviluppo di un
Prodotto Software in ambito PSD2

Relatore

Professor Marco Torchiano

Candidata

Anna Perazzo

Anno Accademico 2019/2020

1 SOMMARIO

ABSTRACT

L'elaborato si propone di analizzare le attività svolte nell'ambito di un progetto di sviluppo di un prodotto software con il fine di comprendere i fattori e le motivazioni che hanno portato a preferire i metodi di gestione prescelti rispetto ad altri disponibili.

Il Progetto che costituisce l'oggetto dell'analisi riguarda lo sviluppo di un prodotto software di uno dei maggiori gruppi bancari italiani.

Il progetto nasce e si sviluppa nella sfera della PSD2. La Payment Service Directive nota come PSD2 è una direttiva Europea (direttiva UE 2015/2366) che ha lo scopo di regolamentare i servizi di pagamento nel mercato europeo disciplinando in particolare, l'uso di nuovi strumenti digitali e servizi di pagamento già in uso.

La PSD2 forza le Banche a rendere accessibili i loro sistemi e consente ai consumatori di condividere i propri dati finanziari, aprendo il mercato allo sviluppo di nuovi servizi e a nuove opportunità di business.

Lo scopo principale della Payment Service Directive è quello della tutela del consumatore accelerando la trasformazione del settore dei servizi finanziari da un universo product centric a universo consumer centric. La PSD2 vuole offrire al consumatore migliori standard in termini di sicurezza, una maggiore convenienza e allo stesso tempo aumentare la qualità e la varietà dei servizi offerti.

Per realizzare i propri obiettivi la normativa pone alle banche nuove sfide tecniche da affrontare che riguardano la sicurezza, l'implementazione di modalità di accesso ai propri sistemi, l'adattamento ad un sistema turbolento ed in continua evoluzione.

Le sfide poste alle banche non sono solo quelle tecniche. Abilitando l'ingresso di nuovi player nel settore e richiedendo la trasparenza dei modelli di pricing adottati dagli operatori finanziari, la normativa incrementa la competizione nel settore e spinge i prezzi verso il basso.

Se da un lato la PSD2 si presenta come una minaccia ai modelli di business tradizionali, dall'altra offre e abilita diverse opportunità di innovazione.

il settore Bancario in un contesto turbolento caratterizzato da rapidi cambiamenti, dove rispondere tempestivamente ai fenomeni innovativi e alle richieste del mercato è fondamentale per restare competitivi.

Al fine di rispondere a queste esigenze è necessario che le aziende adottino modelli di gestione dei processi in grado di farvi fronte. In particolare, per esser competitivi è necessario soddisfare due requisiti:

- ottenere time to market brevi
- soddisfare le esigenze dei consumatori

Si sono quindi analizzati diversi modelli di gestione dello sviluppo software a partire dai primi sviluppati ma ancora oggi utilizzati in determinati contesti fino ai più recenti approcci Agile. Nell'analisi si evidenziano i vantaggi e gli svantaggi di ciascun modello ed in relazione ad essi

suggerisce la tipologia di progetto per cui il modello si ritiene adatto. Non esiste infatti una metodologia che sia in grado di garantire il successo di ogni progetto.

Una volta analizzata l'adattabilità di ciascun modello, si è passati ad illustrare il progetto oggetto di studio, evidenziando le caratteristiche che hanno portato alle scelte adottate nella gestione della progettualità.

L'approccio di gestione utilizzato nell'ambito della progettualità infatti, non è univoco, si è scelto un approccio ibrido, con elementi tipici della metodologia agile e altri di approcci più tradizionali.

Particolare attenzione si è posta alla gestione delle attività di Test, quindi alle attività di Test Management e Defect Management che si sono rivelate le attività più critiche e time consuming. La criticità delle attività di test è legata in particolare alle specificità del contesto PSD2, che hanno richiesto un'attenta e specifica gestione al fine di poter rilasciare un prodotto software il più possibile privo di difetti o malfunzionamenti e rispondente alle aspettative della clientela.

In un contesto consumer centric come la PSD2 è fondamentale per rimanere competitivi realizzare prodotti in grado di soddisfare le sempre crescenti aspettative dei consumatori. Questo assume particolare rilevanza in questo caso, dove il prodotto è rilasciato da una banca, parte di uno dei maggiori gruppi bancari italiani che ha quindi una brand reputation da difendere ed è per questo che garantire elevati standard qualitativi del prodotto è di cruciale importanza.

L'esperienza del caso di studio proposta mostra quindi, come la scelta di un approccio ibrido sia quella che consente di gestire ciascuna fase progettuale in funzione della propria specificità con la metodologia più adatta, mitigando le conseguenze svantaggiose che ciascuna approccio comporta quando è implementato.

1. L'OPEN BANKING

Si definisce "open banking" un modello in cui i dati bancari sono condivisi fra due o più parti non correlate per portare sul mercato migliori competenze. (Laura Brodsky, 2018)

L'Open banking introduce nel mondo del Financial Services logiche di rete, per cui la banca viene intesa come parte di un ecosistema più ampio, fortemente dinamico ed interconnesso, basato sull'interazione e sulla condivisione delle informazioni con soggetti "terze parti".

Per Carlo Marcoli di IBM, l'Open Banking è un concetto associato ad una serie di iniziative che prendono spunto da tre principi fondanti:

1. Il cliente Bancario è il vero proprietario dei propri dati finanziari.

Il cliente bancario che si muove nell'ecosistema dell'Open Banking, non solo ha libero accesso ai propri dati finanziari ma possiede soprattutto, una maggiore libertà e possibilità nella loro disposizione ed utilizzo. Il nuovo consumatore può facilmente consentire l'accesso ai propri dati a parti terze che possono a loro volta, fornirgli una varietà di prodotti e servizi sempre più innovativi e customizzati

2. I prodotti bancari sono trasparenti e facilmente comparabili

L'Open banking rende accessibili dati e informazioni sui prodotti finanziari che diventano quindi più facili da comprendere e comparare.

3. Diverse parti lavorano insieme per creare e produrre migliori prodotti e servizi.

L'Open Banking porta infatti, ad una transizione da modelli di business chiusi a modelli di business aperti collaborativi, in cui servizi, prodotti e dati vengono condivisi fra le parti per la creazione di nuovo valore e nuovi modelli di business. (Deloitte, 2019)

L'Open Banking ha quindi il potenziale di ridisegnare il contesto competitivo e l'esperienza del consumatore del settore bancario.

Ma vi sono anche rischi nascosti, in particolar modo quelli legati alla privacy e i rischi correlati alla trasmissione di dati sensibili.

Diversi sono i fattori che accelerano e contribuiscono l'affermazione del nuovo paradigma nel settore dei Financial Services, fra questi troviamo la Payment Service Directive, l'API Economy e l'affermarsi delle Fintech.

1.1 LA PAYMENT SERVICE DIRECTIVE

La Payment Service Directive nota come PSD2 è una direttiva Europea (direttiva UE 2015/2366) che ha lo scopo di regolamentare i servizi di pagamento nel mercato europeo disciplinando in particolare, l'uso di nuovi strumenti digitali e servizi di pagamento già in uso.

La PSD2 rientra in un trend globale della normativa finanziaria che pone l'enfasi su innovazione e sicurezza. Infatti, gli elementi cardine su cui è costruita sono: Sicurezza, Centralità del Cliente e Innovazione.

La complessa architettura normativa della PSD2 infatti, comprende questioni che vanno dalla trasparenza dei prezzi, alla sicurezza e alla tecnologia. La direttiva comunque, ha offerto ai governi nazionali margini di manovra nell'elaborazione di precisi requisiti.

L'obiettivo del legislatore europeo è quello di garantire una maggior protezione del consumatore e al contempo di consentire lo sviluppo di nuove soluzioni di pagamento, nel perseguimento dell'innovazione. Per far ciò la direttiva non si limita a disciplinare i nuovi player del mercato finanziario e impone la standardizzazione delle infrastrutture per ottenere un miglioramento in termini di efficienza.

La prima richiesta della direttiva è quella di una maggiore trasparenza che si traduce in standard di rendicontazione più rigorosi per le banche, nonché una maggiore trasparenza dei prezzi. La normativa richiede che i prezzi siano non discriminatori, ovvero i costi per l'accesso all'account e l'avvio dei pagamenti devono essere gli stessi per i clienti finali e i terzi.

La normativa prevede inoltre, che parti terze, non solo banche, possano fornire servizi che precedentemente solo le banche erano in grado di offrire.

Le banche infatti, previo consenso del cliente sono obbligate a fornire alle parti terze, l'accesso ai conti correnti dei loro clienti attraverso interfacce facilmente integrabili, come le open-API. Il Third Party Provider può essere un istituto bancario presente sul mercato, una fintech o un'azienda che offre servizi di pagamento, in ogni caso la comunicazione fra le interfacce delle parti deve garantire elevati standard di Sicurezza.

Pe garantire elevati standard di sicurezza, la PSD2 richiede agli operatori finanziari di implementare dei sistemi di autenticazione sicuri e fra loro armonizzati. È necessario che vengano implementati meccanismi di Strong Customer Authentication (SCA) attraverso l'identificazione multi-fattore, che richiede che l'identità dell'utente venga accertata attraverso due o più mezzi di autenticazione.

Consentendo ai Third Party Provider di interfacciarsi con le banche, la PSD2 si pone l'obiettivo di offrire all'utente costi più bassi e una maggiore sicurezza e allo stesso tempo abilita l'introduzione di nuovi business model e la creazione di nuovi prodotti e servizi che consentano di differenziare le esperienze dei consumatori e renderle il più possibile customizzate. La normativa si propone quindi di creare maggiore valore per il consumatore finale, abilitando una maggiore varietà di prodotti e servizi customizzati.

Per introdurre nuovi business model e aumentare la sicurezza nell'accesso ai conti e nei pagamenti on line, la PSD2 è portata a cambiare radicalmente la value chain dei pagamenti e l'uso dei dati nel settore dei Servizi Finanziari.

1.1.1 Timeline

L'8 Ottobre 2015 il Parlamento Europeo ha adottato la proposta della Commissione Europea di creare un mercato dei pagamenti più sicuro e innovativo. Il 16 Novembre 2015 il Consiglio dell'Unione Europea ha approvato la PSD2, garantendo agli Stati Membri un periodo di tempo di due anni per incorporare la direttiva all'interno dei corpi normativi e regolamentativi nazionali.

Il 27 Novembre 2017 la Commissione Europea ha integrato la PSD2 con il Regolamento (EU) 2018/389 che articola e specifica gli standard tecnici di sicurezza e comunicazione in ambito PSD2.

Il 13 gennaio 2018 entra in vigore il Decreto legislativo n.218/2017 di recepimento della PSD2, con l'obiettivo di promuovere lo sviluppo di un mercato dei servizi di pagamento efficiente, sicuro e competitivo.

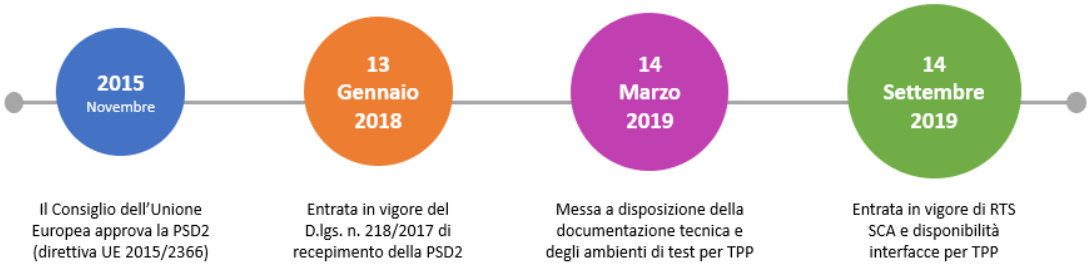


Figure 1. Timeline PSD2

La storia della PSD2 è caratterizzata da ulteriori due milestones.

Il 14 Marzo 2019 cade l'obbligo per tutte le istituzioni finanziarie di mettere a disposizione dei Third Party Provider la documentazione tecnica e gli ambienti per i test.

Il 14 Settembre 2019 infine, gli istituti finanziari hanno l'obbligo di adeguarsi ai Regulatory Technical Standard (RTS)[1]. L'European Bank Authority (EBA)[2] ha concesso un'estensione di 15 mesi, fino a dicembre 2020 per adeguarsi

[1] Regulatory Technical Standards, documento che definisce il dettaglio degli standard tecnici da garantire perché i pagamenti elettronici possano rispettare gli standard di sicurezza previsti dall'Unione Europea a partire da Novembre 2017

[2] L'European Bank Authority è un organismo dell'Unione europea con compiti di sorveglianaza del mercato bancario europeo

1.1.2 I Casi d'uso della PSD2

La PSD2 apre il mercato allo sviluppo di nuovi servizi e a nuove opportunità di business. I principali attori coinvolti nell'esecuzione dei casi d'uso della PSD2 sono tre:

- Payment Service User (PSU): il consumatore finale di una Banca che dispone di un conto attraverso cui può eseguire pagamenti elettronici. E' il proprietario dei propri dati finanziari e può fornire il consenso ad altre organizzazioni ad accedere a questi dati per compiere azioni per suo conto
- Account Servicing Payment Service Provider (ASPSP): la banca di radicamento. Ha la responsabilità dell'esposizione di adeguate interfacce che consentano l'accesso ai dati finanziari dei propri clienti da parte delle parti terze autorizzate. La banca è responsabile dell'esecuzione della Strong Customer Authentication, anche quando l'accesso ai dati è eseguito da parti terze. La banca è anche responsabile di prevenire e mitigare il rischio di frodi
- Third-Party Provider (TPP): organizzazioni che possono offrire servizi con valore aggiunto utilizzando i dati esposti dalle banche. I TPP sono responsabili di richiedere il consenso al consumatore per compiere azioni per suo conto.

Third Party Provider

I Third Party Provider sono dei soggetti autorizzati dalla normativa a fornire servizi di pagamento e nuovi servizi finanziari, abilitati dal consenso fornito dal cliente bancario all'accesso ai propri dati finanziari. I TPP possono operare come Payment Initiation Service Provider (PISP), Account Information Service Provider (AISP) o come Card Issuer Service Provider (CISP).

Per poter operare come TPP un ente deve compiere alcuni passi:

1. Effettuare domanda di autorizzazione per operare come TPP alla Banca d'Italia che entro 90 giorni dal ricevimento della domanda, deve rifiutare o accettare la domanda di autorizzazione
2. Ricevuta l'autorizzazione da parte della Banca d'Italia, il TPP deve registrarsi presso il registro delle imprese
3. I dati sui TPP autorizzati devono essere conservati nel Registro dell'EBA
4. Grazie alle Sandbox di Test, la cui implementazione è prevista dalla normativa, i TPP possono testare i loro servizi con dati fittizi, in sicurezza
5. Per effettuare le operazioni ed i test bisogna assicurare un canale di comunicazione sicuro tra TPP ed istituti finanziari, a tal fine i TPP devono dotarsi di due certificati digitali qualificati nell'ambito dei servizi definiti nel regolamento eiDAS (Electronic Identification, Authentication and Trust Services)[3]
    - Il QWAC (Qualified Web Authentication Certificate) che consente di confermare l'identità e il ruolo del TPP per i clienti e le altre aziende in fase di verifica delle informazioni sensibili

[3] eiDAS è un regolamento europeo in merito all'identificazione elettronica e ai servizi fiduciari per le transazioni elettroniche nel mercato europe. Stabilito nel regolamneto europeo 910/2014 nel luglio 2014

- Il l QsealC è un sigillo elettronico che è un garante dell'integrità e della sicurezza dei dati e della loro origine (PSD2 EIDAS test Certificates (QWAC and QSeal), n.d.)

Payment Initiation Service Provider (PISP)

Immaginiamo il caso di un cliente (PSU) che debba fare un acquisto on line. Una volta giunto alla pagina dei pagamenti l'utente può scegliere come completare il suo acquisto: attraverso una carta di credito o di debito, utilizzando Paypal o "direttamente dal conto corrente". L'utente che sceglie la terza opzione è reindirizzato verso un TPP nel ruolo di PISP. Il PISP richiede il consenso del consumatore per effettuare il pagamento attraverso la banca dell'utente. La banca conferma il pagamento ed il PISP ed invia una notifica all'utente. A questo punto l'e-commerce riceve il pagamento e l'acquisto si conclude.

Figure 2. Schema funzionamento PISP

Con i PISP è possibile bypassare i tradizionali circuiti di pagamento.

Consideriamo ad esempio il processo con cui tradizionalmente avviene il pagamento mediante carta di credito.

La carta di credito viene emessa da un emittente, un ente finanziario o una banca. L'ente esercente è invece il merchant ovvero un esercizio commerciale che consente il pagamento tramite carta o bancomat aderendo ad un circuito di pagamento ed installando un terminale che consenta il pagamento tramite carta. I gestori dei circuiti di pagamento (processor) consentono il trasferimento delle informazioni dall'ente esercente all'ente emittente. I gestori dei terminali a loro volta gestiscono i flussi comunicativi fra gestori dei circuiti e merchant. Ogni volta che un cliente effettua un pagamento tramite carta tutti gli attori del processo interagiscono e devono esser remunerati.

Quando un cliente completa un acquisto in un punto vendita attraverso una carta di credito:

- il terminale invia la richiesta di autorizzazione e i dati della carta al circuito di pagamento
- il gestore del circuito di pagamento inoltra la richiesta all'ente emittente
- l'ente emittente invia l'autorizzazione al circuito di pagamento
- il terminale comunica l'autorizzazione al merchant
- il merchant emette lo scontrino

A fine giornata attraverso dei flussi batch[4], il merchant invia al gestore del terminale i dati relativi alle transazioni della giornata, le informazioni vengono inoltrate al circuito di pagamento che a sua volta le comunica agli enti emittenti competenti, i quali elaborano le informazioni ricevute ed inviano l'estratto conto ai rispettivi titolari delle carte di pagamento.

Grazie al PISP queste operazioni vengono semplificate.

Immaginiamo che il cliente abbia autorizzato il PISP ad accedere ai suoi dati finanziari e a compiere delle operazioni attraverso il proprio conto disponibile on line, in fase di conclusione dell'acquisto il cliente può effettuare il pagamento mediante PISP che può trasferire direttamente l'importo dal conto del cliente al merchant, comunicando con esso attraverso la geolocalizzazione, NFC o QR Code. (KPMG Advisory, 2018)

Figure 3 Applicazione del PISP

Account Information Service Provider (AISP)

Un cliente multi-bancarizzato[5] può consentire ad un TPP di accedere a tutti i suoi conti bancari. La terza parte richiede le informazioni sui conti a ciascuna delle banche, interagendo con le open API delle Banche. Il provider, nel ruolo di AISP, lancia un'applicazione che mostra i saldi ed i movimenti di tutti i conti posseduti dall'utente.

[4] Flussi di dati raccolti in un determinato intervallo temporale

[5] Un cliente multi-bancarizzato è un cliente che possiede diversi controcorrenti a lui intestati, presso differenti gruppi bancari

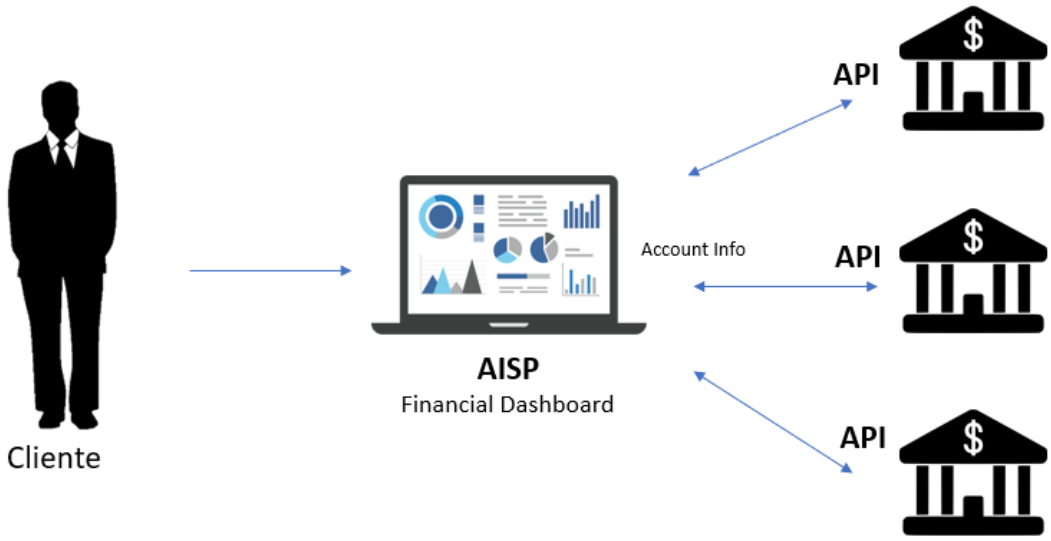


Figure 4 Schema funzionamento AISP

Le informazioni ricevute dalla terza parte che opera come AISP possono costituire la base per l'elaborazione di nuovi prodotti o servizi finanziari ed incrementare l'offerta di valore verso il cliente finale.

Avendo a disposizione l'interezza dei dati finanziari del consumatore, l'AISP ha la capacità di migliorare i processi di profilatura del cliente e diviene quindi in grado di offrire prodotti e servizi sempre più personalizzati. Ad esempio, avendo accesso facilmente ai dati finanziari della propria clientela, gli istituiti di credito possono velocizzare e migliorare i propri processi di scoring.

Secondo una ricerca condotta da KPMG Advisory, I clienti multibancarizzati in Italia nel 2018 sono cresciuti del 63% rispetto all'anno precedente e gli intervistati si rivelano disponibili ad accedere ai propri conti bancari attraverso un unico strumento digitale. (Ylenia Bezza, 2018)

Tale risultato porta a pensare all'esistenza di un mercato, potenzialmente in crescita per i provider che si configurano come AISP.

Card Issuer Service Provider (CISP)

I CISP possono fornire all'utente delle carte di pagamento, pur non gestendo dei conti correnti, ma bensì collegando le carte ai conti correnti posseduti dall'utente.

1.1.3 Realizzazione Casi d'uso della PSD2

L'implementazione dei casi d'uso abilitati dalla PSD2 richiede lo svolgimento di alcune attività preliminari:

1. Registrazione del TPP. Per utilizzare le API delle Banche, un TPP deve fornirsi di certificati eiDAS e richiedere le credenziali alla banca attraverso i Developer Portal messi a

disposizione dalla banca. Il TPP infatti, “chiama” un set di API per iscriversi all’interfaccia di accesso ai conti della banca. La banca quindi valida il certificato eiDAS del TPP e se ne serve per identificare la parte terza. Il TPP viene poi identificato grazie ai dati forniti dal Registro Europeo. Compiute le verifiche il TPP viene registrato come valido per accedere alle API della banca

2. Enrollment dell’utente e consenso. Una volta registrato il TPP chiede alla banca di poter eseguire un’operazione per conto dell’utente. L’utente deve autenticarsi presso la banca di appartenenza e fornire al TPP il consenso, affinché questo possa accedere ai suoi conti. A seguito di ciò il TPP riceve un token di accesso che può utilizzare per compiere delle operazioni per conto dell’utente.
3. Accesso alle API. Il token di accesso ottenuto dal TPP viene utilizzato per effettuare *chiamate alle API.* Il token di accesso viene verificato dalla banca. Quando la verifica si conclude con successo la chiamata arriva fino al gateway interno per accedere alle piattaforme i back end e compiere le operazioni. (Fabrick)

Sono diverse le modalità con cui un operatore finanziario può implementare una soluzione PSD2. La figura di seguito ne illustra un esempio.

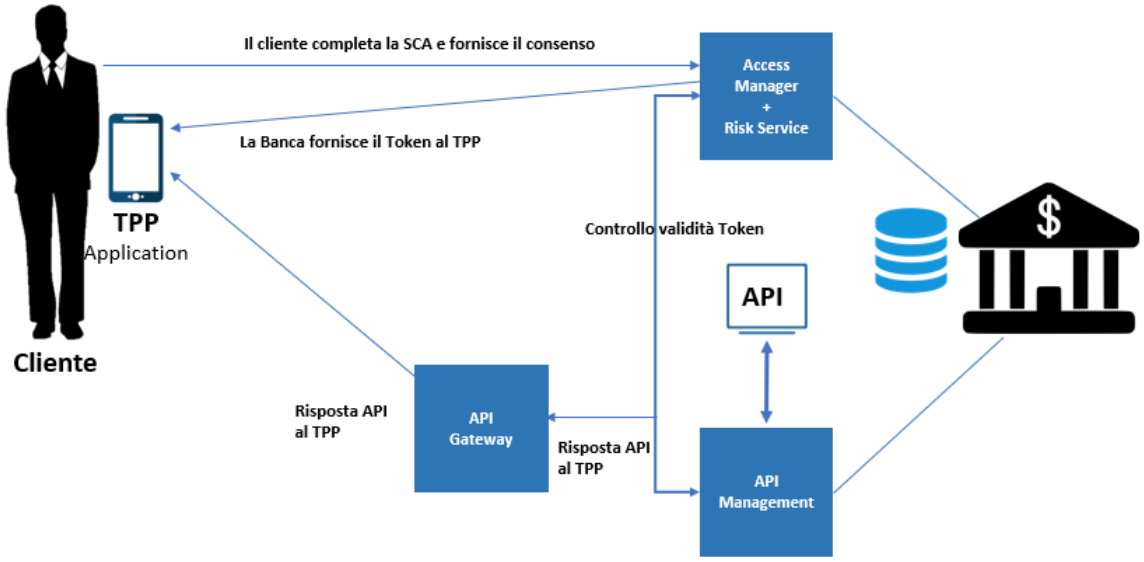


Figure 5 Esempio implementazione caso d'uso PSD2

Immaginiamo che l'utente avvii un'applicazione TPP. Per poter usufruire dei servizi di AISP e PISP è necessario che l'utente colleghi all’applicazione un conto corrente bancario. L'applicazione TPP richiede quindi all'utente di selezionare una banca. Una volta selezionata la banca da collegare, il TPP reindirizza l'utente alla suddetta banca. La banca elabora la richiesta e richiede all'utente di autenticarsi, eventualmente completando una SCA e di fornire il consenso al TPP.

Fornito il consenso e completata la SCA, la banca emette un token di accesso al TPP.

Quando il TPP chiama un'API della Banca con il token di accesso, l'API Gateway verifica attraverso l'access manager se il token di accesso è valido o meno. L'access manager notifica quindi al gateway la validità del token di accesso, il gateway consente l'accesso all'API e la risposta dell'API viene rinviata al TPP. (IBM, 2019)

1.1.4 Architettura IT di un Ente Bancario conforme alla normativa PSD2

La PSD2 forza le banche a rendere accessibili i loro sistemi e consente ai consumatori di condividere i propri dai finanziari. Si propone quindi, come un acceleratore della rivoluzione del mondo dei Financial Services, iniziato con l'Open Banking.

La PSD2 pone nuove sfide tecniche da affrontare per le banche che riguardano la sicurezza, l'integrazione e l'adattamento ad un sistema in continua evoluzione, caratterizzato da un elevato grado di incertezza.

Il requisito tecnologico fondamentale da soddisfare è quello dell'implementazione di un'interfaccia che consenta alle parti terze di accedere ai dati finanziari dei clienti bancari, quale ad esempio un Open Application Programming Interface o API.

Le soluzioni PSD2 richiedono la gestione delle API, soluzioni per la gestione degli accessi con attraverso la Strong Customer Authentication (SCA) e Risk Based Services per rilevare le minacce. Infatti, le banche sono responsabili della mitigazione dei rischi di frode e dovranno implementare controlli avanzati sulle chiamate in arrivo dalle API e strumenti che identifichino possibili frodi. (Marcoli, 2018)

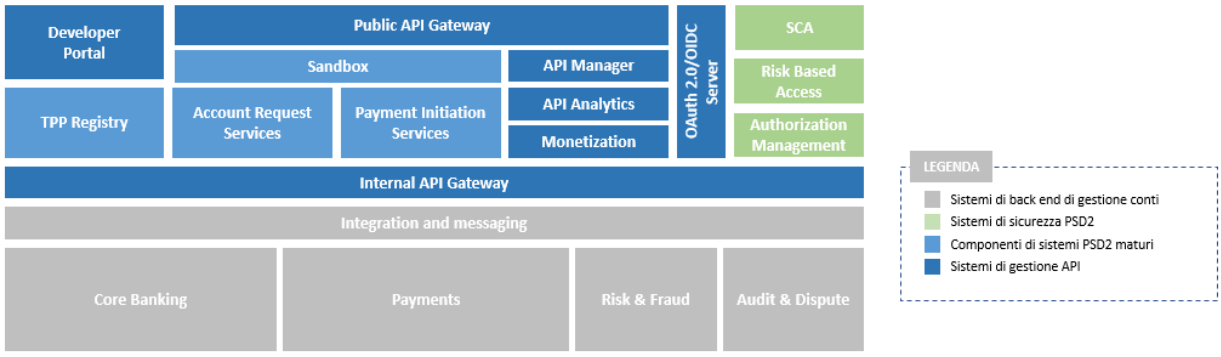


Figure 6 Esempio architettura IT ente bancario conforme alla PSD2

Nella figura sono riportati

1. I sistemi di back-end che gestiscono conti e pagamenti
2. I componenti di sicurezza per la PSD2, quali le componenti necessarie per eseguire la Strong Customer Authentication (SCA), il Risk Based Access e l'Authorization Management per rilevare possibili rischi associati alle chiamate alle API da parte di soggetti terzi
3. i gateway per applicare le politiche di accesso alle API

4. un Developer Portal che consente agli sviluppatori TPP di iscriversi ad API
5. l'API manager e l'API Analytics per controllare il ciclo di vita delle API

Queste componenti non sono tutte obbligatorie, ma diventano essenziali se la banca vuole raggiungere obiettivi che vanno oltre i requisiti di conformità.

Altri componenti possono essere aggiunti in base alla maturità della strategia API: il registro TPP contiene una copia dei dati memorizzati nel registro europeo, la Sandbox invece, consente ai TPP di effettuare i test con dati fittizi e procedere allo sviluppo continuo di servizi che utilizzano l'interfaccia della banca.

1.1.5 Tecnologie abilitanti: le API

Le API sono un set di definizioni e protocolli con cui vengono realizzati o integrati dei software applicativi. Le API o Web API abilitano le interazioni consentite dalla PSD2 fornendo un canale di comunicazione sicuro e facilmente accessibile. Le API consentono di scambiare dati e funzionalità in maniera efficiente e flessibile. (Red Hat)

Le API non vanno intese semplicemente come un mero strumento tecnico che consente la condivisione dei dati. Possono infatti diventare una leva strategica di una impresa. Infatti, permettendo alle organizzazioni di condividere dati e servizi, abilitano innovazioni rapidi e aprono il settore a nuovi beni e servizi.

Le Open API (nate nel 2010 con il nome di Specifica Swagger) consentono a sviluppatori esterni rispetto ad un'organizzazione di accedere ai dati di back end e di utilizzarli per proporre nuovi prodotti o servizi o migliorare quelli esistenti.

La possibilità delle API di generare nuovo valore, consentendo rapide innovazioni di prodotto e servizio, abilitando la nascita di piattaforme e di nuovi modelli di business ha portato gli esperti di discutere di API Economy.

Le API sono uno dei principali fattori abilitanti la trasformazione del settore bancario da un sistema chiuso ad un sistema aperto. Condividendo i dati attraverso le interfacce delle API si instaura fra gli attori del settore un modello collaborativo che porta ad innovazioni in termini di nuovi prodotti e business model con time to market ridotti. La collaborazione fra stakeholder trasforma il sistema finanziario in un ecosistema che è in grado di evolvere rapidamente e rapidamente comprendere e rispondere alle esigenze del mercato. La rapida comprensione e risposta sono facilitate dalle partnership fra enti che possiedono diverse competenze tecnologiche e finanziarie.

L'esposizione delle API comporta tuttavia dei rischi in termini di sicurezza che vanno valutati e mitigati. È per questo motivo che la normativa interviene per garantire la sicurezza del consumatore imponendo elevati standard di sicurezza. Il fatto che le API siano esposte non implica che tutti possano avere accesso, come per i casi d'uso della PSD2 è necessario che le parti terze abbiano l'autorizzazione ad accedere.

1.1.6 Sicurezza: Strong Customer Authentication

Gli utenti che compiranno le operazioni in ambito PSD2, tipicamente, inizieranno le proprie operazioni interagendo con l'interfaccia utente del TPP.

Quando il TPP invierà richiesta di accesso ai conti dell'utente verso Account Servicing Payment Service Provider (ASPSP), l'utente dovrà eseguire una Strong Customer Authentication e quindi autenticarsi presso l'ASPSP e dimostrare di aver fornito il consenso per le operazioni che il TPP richiede di eseguire.

Il requisito della Strong Customer Authentication risulta particolarmente sfidante dal punto di vista della User Experience. Nel processo infatti, non sono più solo due le parti coinvolte ma tre: l'utente, la sua banca e il TPP. Il Customer Journey inizia e termina, invece, con l'interfaccia del TPP.

Il TPP comunicherà con l'ASPSP attraverso le Open API. Diversi enti regolatori hanno rilasciato delle indicazioni su come implementare la SCA.

Il 23 Febbraio l'European Bank Authority ha pubblicato la bozza finale dei Regulatory Technical Standards (RTS) riguardo alla Strong Customer Authentication e la sicurezza delle comunicazioni in ambito PSD2.

Lo standard Berlin Group[6], adottato in molti paesi Europei fra cui l'Italia, prevede tre differenti modalità di implementazione della SCA: SCA Redirect, SCA Embedded, SCA Decoupled.

Nell'ambito della PSD2 la SCA consiste nel verificare 3 fattori: un fattore di conoscenza (il PIN, ad esempio), un fattore di inerenza (riconoscimento facciale) e un fattore di possesso (un dispositivo). È necessario garantire l'autenticazione attraverso il riconoscimento di almeno due dei fattori considerati. (PwC Italia)

1.1.6.1 SCA Redirect

Il modello Redirect prevede che l'Utente inizi ad interagire con il TPP e venga re-direzionato verso l'interfaccia web o l'App mobile della propria banca, dove effettua l'autenticazione.

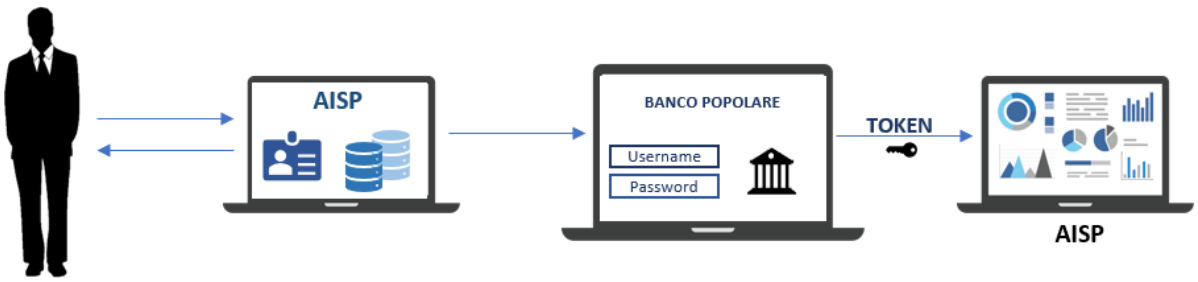


Figure 7 SCA Redirect

[6] Il Berlin Group è un organo europeo di standardizzazione tecnica che opera riguardo all'interoperabilità dei sistemi di pagamento. Ad esso hanno aderito 26 fra i maggiori nell'industria dei pagamenti europea.

In questo modello è la banca a gestire il processo di autenticazione dell'utente in modo autonomo. Quindi le Open API utilizzate dal TPP per interfacciarsi con la Banca non sono coinvolte nelle operazioni di gestione della SCA.

Il modello Redirect presenta alcuni vantaggi:

- la banca ha il pieno controllo del processo di autenticazione dell'utente. Se quindi ha già implementato una soluzione, potrebbe riutilizzarla se questa è in linea con gli standard definiti nell'RTS
- la banca implementare la propria soluzione di SCA come parte del proprio piano di adeguamento alla normativa, senza dipendere da parti terze
- l'utente potrebbe sentirsi più a proprio agio e sicuro autenticandosi dall'interfaccia della propria Banca (Fido Alliance, 2018)

1.1.6.2 SCA Decoupled

Il modello Decoupled prevede che l'utente si autentichi attraverso l'app mobile della propria banca, o qualsiasi altra applicazione dedicata all'autenticazione che sia indipendente dal Front End dell'Internet Banking.

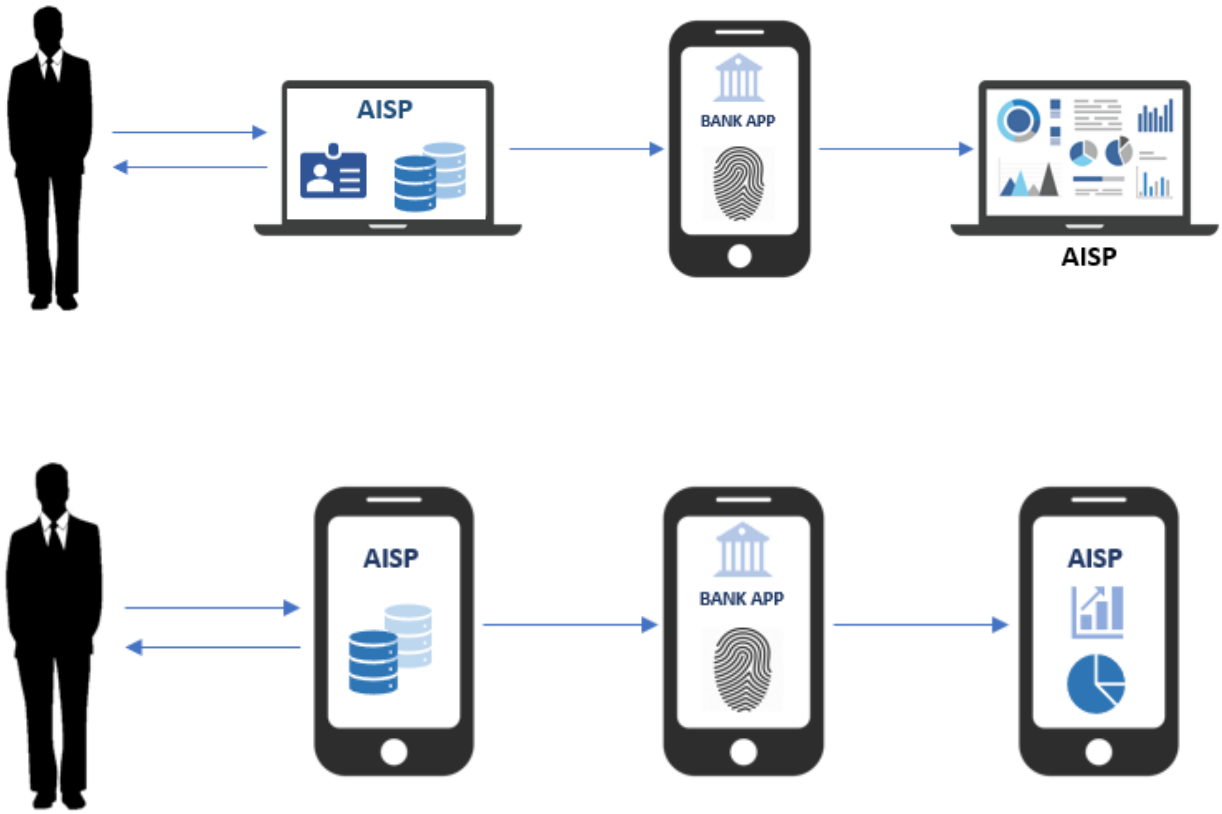


Figure 8 SCA Decoupled

Il modello Decoupled migliora la User Experience: quando l'utente inizia ad interagire con il TPP attraverso l'interfaccia web del TPP, rimarrà sulla sua interfaccia web. L'autenticazione avviene attraverso la Mobile App della banca.

Il prerequisito per l'implementazione di questa modalità di SCA è che l'utente possieda uno smartphone attraverso cui può fornire il consenso utilizzando la funzionalità di SCA della Mobile App della propria banca. Poiché non tutti li utenti possiedono uno smartphone l'approccio Decoupled non può essere utilizzato da solo. (Fido Alliance, 2018)

(FIDO and PSD2: Solving the Strong Customer Authentication Challenge in Europe, 2018)

1.1.6.3 SCA Embedded

Quando si applica il modello Embedded, la SCA dell'utente è eseguita interamente attraverso l'interfaccia utente del TPP. Il modello Embedded migliora notevolmente la user experience ma allo stesso tempo, presenta molte difficoltà, ad iniziare dallo step della User Verification.

Figure 9 SCA Embedded

Nel modello Embedded, la conoscenza del fattore di inerenza potrebbe esser verificata dalla Banca ma, se dovesse esser fatto attraverso l'interfaccia del TPP, richiederebbe la trasmissione dei dati dal TPP alla Banca. Il che potrebbe introdurre nel modello molteplici difficoltà: relativi allo storage e alla trasmissione di dati sensibili, ma anche allo user journey. Il possesso del dispositivo di autenticazione nel modello embedded potrebbe esser verificato dalla Banca in differenti modi: ad esempio, la Banca potrebbe inviare un codice numerico come un OTP (One Time Password) al dispositivo dell'utente che una volta visionato il codice sul proprio device lo digiterebbe nell'apposito campo previsto dall'interfaccia del TPP e il TPP lo inoltrerebbe alla Banca per la verifica. In questo modo il canale di comunicazione attraverso cui si passa l'OTP è indipendente dal TPP.

(Fido Alliance, 2018)

(FIDO and PSD2: Solving the Strong Customer Authentication Challenge in Europe, 2018)

1.2 LA BANCA DIGITALE

All'interno del settore dei servizi finanziari è quindi in corso una vera e propria rivoluzione, destinata segnare profondamente il modo di operare delle imprese nel settore ma anche il modo in cui i consumatori si approcciano ai servizi finanziari. L'impatto della PSD2 è disruptive. Ma le radici della rivoluzione risalgono a ben prima dell'entrata in vigore della normativa.

Due sono i principali driver che hanno portato e guidano la rivoluzione del settore finanziario:

- l'entrata di nuovi player nel settore
- le aspettative dei clienti digitali

I nuovi player e "nuovi" clienti sono spinti entrambi dalla disponibilità di nuove tecnologie. A dare forma alle aspettative dei nuovi clienti è infatti la trasformazione digitale che attraverso tutti i settori industriali ed è sempre la tecnologia ad abilitare enti ad operare nel settore dei servizi finanziari, enti come le Fintech o i digital Champions quali Google ed Amazon.

La PSD2 infatti si pone l'obiettivo di regolamentare le operazioni dei nuovi player del mercato, facilitandone allo stesso tempo l'accesso e pone il consumatore al centro del testo normativo. La PSD2 vuole offrire al consumatore migliori standard di sicurezza, una maggiore convenienza e allo stesso tempo aumentare la qualità la varietà dei servizi offerti.

La trasparenza in termini di pricing richiesta dalla normativa spinge verso il basso i prezzi e la varietà e pluralità di servizi offerti aumentano il potere contrattuale dei clienti bancari che scelgono sempre più spesso la banca on line per ragioni di convenienza economica.

La PSD2 abilita i Third Party Provider ad operare per conto dei clienti bancari, accedendo ai loro dati finanziari. A questa intermediazione è associato un forte rischio per le banche, ovvero quello di perdere il ruolo di riferimento finanziario per i propri clienti che potrebbe portare ad una svalorizzazione del rapporto fiduciario. È necessario quindi che le banche pianifichino e attuino opportune strategie per trattenere la propria customer base e queste strategie non possono che passare attraverso l'innovazione.

1.2.1 **I nuovi clienti digitali.**

Le innovazioni in ambito tecnologico hanno modificato sensibilmente le aspettative dei clienti bancari che si aspettano una user-experience sempre più intuitiva ed immediata e soprattutto una maggiore sicurezza e convenienza dei servizi offerti.

Le aspettative dei clienti si formano attraverso le loro interazioni al di fuori del settore bancario che li portano a desiderare sempre più la qualità che gli viene offerta da altri settori che hanno una maggior attenzione verso la customer experience.

Secondo Alberto Dalmasso, CEO di Satispay Italia "la Customer Experience rappresenterà il principale driver dei servizi finanziari per il futuro" …" La chiave di successo sta nel presidio della relazione con il cliente. I clienti, infatti oggi chiedono semplicità di utilizzo immediatezza e una profonda comprensione delle proprie esigenze". (Ylenia Bezza, 2018)

Ma le aspettative dei clienti non riguardano solo la user experience. Tecnologie innovative, quali pagamenti digitali e applicazioni di tipo cloud based aumentano le aspettative dei consumatori in termini di sicurezza, rapidità e convenienza. Nuove tecnologie hanno consentito di aumentare l'offerta di metodi di pagamento rapidi e di utilizzare metodi di pagamento (carte di credito, e-wallet) differenti parallelamente, sulle stesse piattaforme. L'entrata di altri operatori come Fintech nel settore ha poi spinto i prezzi al ribasso.

I risultati dell'indagine del 2018 condotta da KPMG sul Digital Banking hanno rivelato che il numero di Clienti ad utilizzare i Canali di Internet Banking (98%) e Mobile Banking (84%) è in crescita rispetto all'anno precedente, rispettivamente +2% e +6%. Il Mobile banking è il canale il cui utilizzo è caratterizzato dalla crescita più rapida. Tuttavia, il canale Internet Banking e la filiale rimangono i canali preferiti dagli italiani per la gestione delle proprie finanze. (Ylenia Bezza, 2018)

La banca tradizionale rimane il punto di riferimento per la popolazione italiana ma le banche on line stanno progressivamente aumentando le proprie quote di mercato grazie, soprattutto, all'offerta di condizioni economiche più vantaggiose.

Al contempo cresce il numero di Clienti multi-bancarizzati (+63%).

Il cliente italiano è quindi un cliente multi-bancarizzato e multicanale, utilizza infatti tutti i canali messi a disposizione dalla propria banca di radicamento a seconda del tipo di operazione che si vuole alla complessità delle informazioni ricercate e del contesto in cui l'operazione è eseguita. Il Mobile Banking è ad esempio utilizzato quando vi è la necessità di effettuare operazioni in tempo reale. Invece, gli italiani preferiscono recarsi in filiale in tutti i casi in cui si ha bisogno di una consulenza o di informazioni su temi e scelte finanziarie complesse.

Cosa vogliono i clienti dai provider di servizi finanziari durante la loro esperienza di pagamento?

- Facilità d'uso: un'esperienza di pagamento fluida
- Disponibilità: essere in grado di pagare qualsiasi cosa, ovunque si trovi ed in ogni istante
- Sicurezza: essere sicuri dei propri pagamenti e della propria identità
- Trasparenza: comprendere le implicazioni del proprio acquisto
- Risultati istantanei: acquisti completati in tempo reali
- Convergenza: ottenere un valore aggiunto rendendo gli acquisti rimborsabili in futuro (Tuomas Neonen)

Come può la Banca, con la sua architettura IT garantire tutto questo?

Migliorando la user-experience dei propri canali, aumentando l'accessibilità ai servizi clienti la banca può offrire una sempre maggiore facilità d'uso ai propri clienti.

Il fulcro attorno a cui la PSD2 si sviluppa infatti, è il cliente finale, a cui viene reso sempre più facile e sicuro l'acquisto di prodotti on line, ma rende anche disponibili nuovi servizi in grado di gestire i conti bancari e offrire servizi personalizzati.

Con la nuova normativa cambia radicalmente l'esperienza dei consumatori e cresce il ruolo e l'importanza della Customer experience. La PSD2 infatti, si muove quindi nella direzione di una maggiore facilità d'uso e una crescente personalizzazione dei servizi.

1.2.2 **L'ingresso di nuovi player nel settore**.

La rivoluzione digitale dei servizi finanziari causata ha trasformato il settore bancario in un contesto sempre più competitivo consentendo l'ingresso nel mercato di nuovi attori dalle Fintech alle grandi aziende digitali (Facebook, Google, Amazon).

La PSD2 infatti nasce in questo contesto con l'obiettivo di regolamentare la competizione ed i comportamenti di nuovi player nel settore.

La normativa ha consentito a nuovi player di competere nel settore dei pagamenti digitali dominato dalle banche e dai tradizionali provider di servizi finanziari, offrendo ai nuovi player la possibilità di configurarsi come Third Party Provider e accedere alle informazioni dei conti correnti dei consumatori.

Tra i nuovi player troviamo le Fintech. Le Fintech sono fra gli attori principali della rivoluzione del settore finanziario. Le prime Fintech iniziano a fare il loro ingresso nel settore finanziario dopo la crisi del 2007. Utilizzando le competenze tecnologiche a loro disposizione, si propongono di adottare un approccio innovativo, differente, basato su un misto di competenze digitali e finanziarie per dare nuova vita ad un settore legato fortemente ad approcci più tradizionali.

Se al loro ingresso le Fintech si erano affermate essenzialmente come competitor di Banche ed Istituti finanziari tradizionali, ben presto si sono instaurati dei modelli di cooperazione fra gli incumbent ed i new entrant.

Le Fintech infatti, possono offrire servizi sostitutivi o complementari a quelli offerti dagli operatori tradizionali. Lanciano prodotti costruiti su specifiche esigenze del cliente, ottenute studiando la catena di valore del sistema bancario e scovando i bottle-neck e i punti deboli a cui porre rimedio. Prodotti e servizi lanciati dalle Fintech, sono altamente specifici, il loro obiettivo è quello di riempire buchi di offerta, soddisfare specifiche esigenze, risolvere determinati problemi o migliorare processi inefficienti. Focalizzandosi su un solo aspetto della value chain o su uno specifico segmento di mercato, sono in grado di comprendere al meglio i bisogni dei consumatori e disegnare una customer experience fluida che soddisfi le aspettative.

Le Fintech sono organizzazioni agili, il che gli consente di rispondere rapidamente alle richieste del mercato ed innovare rapidamente. Utilizzano infatti approcci Agile nella gestione dei processi di sviluppo, basati su cicli iterativi con la continua verifica e validazione del prodotto da parte dell'utente finale. Approcci di questo tipo consentono di realizzare prodotti rispondenti ai bisogni dell'utente e che garantiscono una migliore customer experience.

1.2.3 La Banca nella PSD2

Se da un lato la PSD2 si presenta come una minaccia ai modelli di business tradizionali, dall'altra offre e abilita diverse opportunità di innovazione. La possibilità di operare come AISP e PISP non è infatti prerogativa esclusiva dei Third Party Provider; le stesse banche possono adottare i nuovi business model abilitati dalla normativa e offrire servizi e prodotti complementari a quelli offerti.

Nuovi prodotti e nuovi modelli di business possono costituire un'opportunità per le banche che in questo modo possono intensificare il rapporto fiduciario con i propri consumatori ma anche attirare nuovi clienti guadagnare quote di mercato.

D'altro canto, la PSD2 abbassa le barriere all'ingresso del settore e Fintech o altri enti possono proporsi come TPP e proporre essi stessi nuovi modelli di business, prodotti, servizi e guadagnare quote di mercato. L'aumento della competizione nel settore causato dai nuovi entranti potrebbe

portare le Banche a vedere ridotte non solo le proprie quote di mercato, ma anche la fiducia che i consumatori ripongono in esse.

Secondo una ricerca del McKinsey Group, per sfruttare al meglio le occasioni offerte dalla PSD2, una piccola banca dovrebbe adottare una strategia di implementazione di una soluzione integrata di pagamenti e management finanziario, mentre una grande banca dovrebbe costruire un suo ecosistema che offra accesso ad un'ampia selezione di applicazioni da diversi provider. Oppure, in una differente ottica, le banche dovrebbero progettare un'architettura tecnologica altamente efficiente in gradi di sostenere soluzioni innovative. (Alessio Botta, n.d.)

Differenti quindi, sono i modi in cui le Banche possono rispondere alla PSD2.

Secondo un'analisi della società di Consulenza PwC condotta sullo storico dei propri Clienti (dati 2014/2016) sono 4 i modi principali con cui una Banca può dare la sua risposta alla normativa e alle nuove esigenze che questa stimola nel consumatore. La Banca può quindi comportarsi come un Compliant Player, Aggregatore, Piattaforma, Aggregatore come piattaforma. (PwC Italia)

1. **Essere Compliant**: è l'obiettivo minimo di ogni banca, quello di conformarsi agli standard previsti dalla normativa. Gli adeguamenti non riguardano solo la tecnologia ma anche aspetti legali e commerciali. Limitarsi al solo soddisfacimento dei requisiti imposti dalla normativa può risultare pericoloso. Il rischio è quello di non essere più competitivi e perdere la propria customer base in un mercato in rapida evoluzione, dove le barriere all'ingresso per i new entrant si sono abbassate e le aspettative dei consumatori sono sempre più alte
2. **L'Aggregatore**. Una delle opportunità che la normativa offre alle Banche è quella di configurarsi come AISP, Account Information Service Provider. L'AISP è un Third Party Provider che consente di collegare ad un'unica applicazione, mobile o software based, i conti correnti di differenti banche da parte di un unico cliente multi-bancarizzato. Il caso d'uso reso disponibile dalla PSD2 offre alle banche la possibilità di disporre di un numero maggiore di dati sui comportamenti finanziari della propria clientela, abilitando quindi l'offerta di prodotti finanziari sempre più customizzati. La maggiore personalizzazione dei servizi proposti potrebbe consentire alle banche di difendere la propria customer base ma anche di aumentare la quota di investimento dei propri clienti multibancarizzati nei loro prodotti
3. **Piattaforma**. La Banca che si pone come piattaforma è in grado di offrire alla propria clientela un insieme di prodotti e servizi customizzati. Il futuro modello di business della banca sarà quindi una piattaforma integrata di idee, processi, tecnologia e informazione (Deloitte), darà accesso a servizi a valore aggiunto che riguardano ogni aspetto dei comportamenti finanziari dei propri clienti. Fondamentale per realizzare una piattaforma di successo è porre al centro del proprio business model il consumatore, investire in un'architettura IT basata sulle API che abiliti lo sviluppo rapido nuovi prodotti e servizi, garantire elevati standard di sicurezza e servirsi di tecnologie che consentano di migliorare la user experience, come Analytics, Artificial Intelligence, ma anche che consentano una riduzione dei costi ed una ottimizzazione delle infrastrutture, quali il cloud (Ylenia Bezza, 2018)
4. **Aggregatore come Piattaforma**. E' l'ente bancario che è in grado di cogliere appieno le opportunità di innovazione offerte dalla normativa. La Banca è in grado di utilizzare le proprie competenze tecnologiche e digitali per offrire ai propri clienti servizi e prodotti sempre più customizzati e in grado di soddisfare ogni esigenza del consumatore correlata in qualche modo al mondo dei servizi finanziari. Essenziale per implementare questa strategia

è la costruzione di partnership con le parti terze, quali le Fintech che consentono di diminuire i costi di sviluppo e innovazione e migliorare il go to market.

La scelta della strategia da attuare richiede un'attenta analisi e valutazione in termini di costi e benefici della loro implementazione, considerando la dimensione dell'ente bancario, l'ammontare degli investimenti necessari, ma anche gli obiettivi di breve e lungo termine della banca. Obiettivi che vanno valutati, soprattutto in funzione della clientela target di ciascuna banca. Nei business model customer centric come quelli abilitati dalla PSD2è essenziale per le banche conoscere i propri clienti per comprendere e soddisfare le loro esigenze.

2 I METODI PER GESTIRE L'INNOVAZIONE

Le innovazioni introdotte e abilitate dalla Rivoluzione Digitale hanno trasformato il settore Bancario in un contesto turbolento caratterizzato da rapidi cambiamenti, dove rispondere tempestivamente ai fenomeni innovativi e alle richieste del mercato è fondamentale per restare competitivi.

Al fine di rispondere a queste esigenze è necessario che le aziende adottino modelli di gestione dei processi in grado di farvi fronte. In particolare, per esser competitivi è necessario soddisfare due requisiti:

- ottenere time to market brevi
- soddisfare le esigenze dei consumatori

Quali metodi di gestione sono in grado di garantire time to market brevi? Quali sono in grado di portare allo sviluppo di un prodotto che soddisfi al meglio le esigenze del cliente? Quali sono in grado di rispondere alle frequenti necessità di cambiamento dei contesti innovativi? Quali sono in grado di garantire flessibilità e adattabilità mantenendo i costi contenuti?

Qual è dunque il metodo miglior per portare avanti un progetto di sviluppo di un prodotto software? Agile? Scrum? A cascata?

La risposta degli esperti sembra non indicare un vincitore. I progettisti sottolineano che la metodologia appropriata da adottare è strettamente legata al contesto all'interno del quale avviene lo sviluppo prodotto e dal prodotto stesso.

La risposta lascia molti insoddisfatti in quanto non fornisce indicazioni o linee guida ed è per questo che recentemente due iniziative internazionali si sono poste l'obiettivo di dare una risposta più precisa alla domanda. Una di queste è la Semat la Software Engineering Method and Theory. Si tratta di un'iniziativa lanciata Ivar Jacobson, Bertrand Meyer, and Richard Soley nel 2009 con lo scopo di sviluppare e promuovere standard internazionali nell'ambito dell'ingegneria del software. Vi è poi il "New Deal", un Manifesto, firmato da esperti del settore che non propone una standardizzazione di metodologie e processi, ma invece si propone di promuovere l'adozione di un una certa flessibilità nella scelta del modello da adottare. Il Manifesto sostiene sia necessario essere più pragmatici nella scelta e adozione di uno specifico modello. In particolare, il Manifesto si rivolge a coloro che assumono un atteggiamento quasi fideistico nei confronti di un determinato approccio,

applicandolo fedelmente in ogni sua parte e li invita a considerare che la metodologia dipende dal contesto in cui avviene lo sviluppo e dal prodotto software stesso e che questi possono variare. Le metodologie adottate devono quindi adattarsi ai cambiamenti. (Comai, 2013)

Di seguito si analizzeranno diversi modelli di gestione dello sviluppo software a partire dai primi sviluppati ma ancora oggi utilizzati in determinati contesti, fino ai più recenti approcci Agile. Nell'analisi si evidenziano i vantaggi e gli svantaggi di ciascun modello ed in relazione ad essi suggerisce la tipologia di progetto per cui il modello si ritiene adatto.

In seguito, le considerazioni svolte verranno applicate ad un caso di studio.

2.1 MODELLI DI GESTIONE DI SVILUPPO SOFTWARE

L'ingegneria del software si basa su un approccio sistematico che utilizza strumenti e tecniche appropriate, opera con determinati vincoli di sviluppo e segue un processo.

Un processo di sviluppo software descrive le attività necessarie allo sviluppo di un prodotto software.

Il processo di sviluppo software è infatti, caratterizzato dalla presenza di diverse fasi normalmente chiamate fasi del software.

- Ingegneria dei Requisiti
- Design
- Implementazione
- Verifica e Convalida
- Manutenzione

Le differenti fasi sono declinate in maniera diversa nei vari modelli di processo di sviluppo software.

Ingegneria dei Requisiti

È il processo che consente di stabilire quali siano le necessità degli stakeholder che il prodotto software deve soddisfare. I requisiti del sistema sono raccolti analizzando le necessità dell'utente.

Come si definiscono i requisiti?

1. Raccolta dei requisiti dagli stakeholder. L'attività può esser condotta in diversi modi: interviste, questionari, analisi di documenti, osservazione, prototipi, casi di utilizzo e schemi statici e dinamici elaborati assieme gli utenti. Fra le maggiori difficoltà riscontrate in questa fase si possono annoverare: requisiti poco chiari, difficoltà ad ottenere requisiti, difficoltà a comprendere i requisiti e tradurre tali requisiti in specifiche progettuali
2. Analisi dei requisiti: studio e comprensione dei requisiti identificati
3. Specifica dei requisiti: i requisiti raccolti vengono rappresentati e organizzati per poi esser condivisi con gli stakeholder del progetto

4. Convalida: bisogna assicurarsi che i requisiti siano completi, consistenti, non ridondanti e che rispettino una serie di caratteristiche necessarie.
5. Gestione dei requisiti: i requisiti possono cambiare durante il ciclo di vita di un progetto, pertanto è necessario prevedere una serie di attività che siano in grado di garantirne l'effettiva gestione.

L'ingegneria dei requisiti ha un'importanza fondamentale nel processo di sviluppo di un prodotto software. Diverse sono le motivazioni, una delle principali risiede nella possibilità di correggere gli errori.

In generale, il costo associato alla correzione degli errori dipende dal numero delle decisioni successive basate su di esso. Quindi un errore nella comprensione dei requisiti ha impatti su ciascuna delle fasi successive ed un costo elevato.

Design

È la fase in cui i requisiti definiti nella fase iniziale del processo vengono analizzati al fine di ottenere una descrizione della struttura e dell'organizzazione interna del sistema. La descrizione è propedeutica alla effettiva implementazione del sistema.

Tradizionalmente, la fase di software design consiste in una serie di attività di progettazione quali: design architetturale, abstract specification, design delle interfacce, design delle componenti, struttura dei dati, design dell'algoritmo.

La divisione in attività può essere realizzata in modi differenti; ma l'idea alla base di ciascuna divisione è che si parte da una visione di ampio spettro del sistema o high level design (HLD), quale la progettazione dell'architettura ad una visione di basso livello del sistema o low level design (LLD) come la progettazione dell'algoritmo. Nell'HLD si definisce la struttura del sistema in termini di componenti e relazioni tra di essi, nell'LLD si definisce invece la struttura interna di ciascun componente.

Implementazione

Dopo aver progettato il sistema, si passa alla sua implementazione. Il modo in cui il software viene costruito si basa su 4 pilastri:

1. Riduzione della complessità: costruire un software che sia facile da comprendere e da utilizzare
2. Anticipazione della diversità: la costruzione del software potrebbe variare in modi diversi nel tempo, in molti casi, in modi inaspettati. Bisogna quindi essere in grado di anticipare alcuni cambiamenti
3. Strutturare per la validazione o per la testabilità: costruire un prodotto software che sia facilmente testabile e verificabile durante la fase di verifica e validazione

4. Conformità con gli standard esterni ed interni

Verifica e Convalida

È la fase dello sviluppo software che ha lo scopo di verificare che il software incontri le specifiche e risponda ai suoi propositi. In particolare, la validazione risponde alla domanda: è stato implementato il sistema corretto? Il sistema risponde alle esigenze e alle necessità dei clienti?

La verifica, invece, deve essere in grado di rispondere alla domanda: il sistema software è stato costruito correttamente?

Manutenzione

Una volta che il prodotto software viene rilasciato e consegnato all'utente, possono accadere molte cose. Potrebbe ad esempio cambiare l'ambiente oppure gli utenti potrebbero aver trovare dei bug nel prodotto software. Per questo, la fase di mantenimento è una fase necessaria dello sviluppo software.

Ci sono tre principali attività di manutenzione:

- Manutenzione correttiva: elimina i problemi con il codice
- Manutenzione di perfezionamento: risponde alle richieste di nuove funzionalità
- Manutenzione adattiva: che si prende cura dei cambiamenti ambientali

Si tratta di un'attività molto importante e spesso costosa.

Un'altra attività costosa è quella di Regression testing. Il test di non regressione deve essere effettuato ogni volta che il software viene modificato per verificare che il software funzioni come ci si aspetta e che i cambiamenti introdotti non comportino effetti imprevisti.

Come bisogna mettere assieme queste attività per sviluppare un software?

A descriverlo o prescriverlo, ci sono i processi o modelli di sviluppo software. Il principale scopo di questi modelli è quello di descrivere l'ordine di esecuzione delle differenti attività, cioè quale attività dovrebbe essere eseguita per prima e quale dovrebbe seguire.

Un altro scopo fondamentale dei processi di sviluppo software è determinare il criterio di transizione fra le differenti attività e cioè quando si può passare da una fase a quella successiva.

2.2 MODELLI LINEARI

I modelli lineari sono modelli sequenziali, adottano un approccio strutturato e sono fortemente legati alla pianificazione (plan-driven).

Il modello a cascata è storicamente il primo modello di gestione del ciclo di vita del software. Nasce con l'obiettivo di descrivere il modo in cui venivano svolte le attività di sviluppo software negli anni '60-'70. L'approccio alle attività di sviluppo prodotto del software era stato mutuato dall'industria manifatturiera, adottandone i principi e i metodi tayloristici. In effetti, il modello a cascata si traduce in una sequenza rigida di fasi che ricorda quelle della catena di montaggio. La rigidità del modello fu ben presto criticata e spinse progettisti e sviluppatori a cercare approcci più flessibili.

Tuttavia, il modello a cascata sebbene modificato con l'introduzione di logiche che ne diminuiscano la rigidità, è ancora oggi utilizzato in progetti di grandi dimensioni ed in progetti strutturati ed organizzati.

2.2.1 Il modello a cascata

Nel 1956 Herbert D. Benington in occasione di un Simposio sui metodi di programmazione descrisse il processo di sviluppo software come un processo diviso in fasi. Nel 1970, invece, Winston W. Royce in un articolo fornì una descrizione formale del modello a cascata. Lo scopo dell'articolo di Royce non era quello di fornire delle indicazioni per completare con successo un processo di sviluppo software ma piuttosto, si poneva l'obiettivo di fornire una descrizione delle pratiche di sviluppo software adottate in quegli anni ed in effetti ne evidenziava i limiti e lo indicava come un modello imperfetto. In tal senso il modello a cascata non è un modello prescrittivo, bensì descrittivo. Nonostante i suoi limiti si tratta di un modello ampiamente utilizzato in quegli anni: nel 1985 il Dipartimento della Difesa degli Stati Uniti ha definito il modello come standard per lavorare con gli appaltatori di sviluppo software.

Il processo di realizzazione del software che segue il modello a cascata o Waterfall è strutturato in una sequenza lineare di fasi, il cui flusso è unidirezionale, come nel caso di una catena di montaggio, per cui non si può dare inizio alla fase successiva se prima non si è completata la fase che la precede. Alla fine di ciascuna fase infatti, è prevista una revisione per determinare se il progetto è pronto per avanzare alla fase successiva. I risultati di una fase sono il punto di partenza e l'input della fase successiva e non possono influenzare una fase precedente. (Royce, 1970)

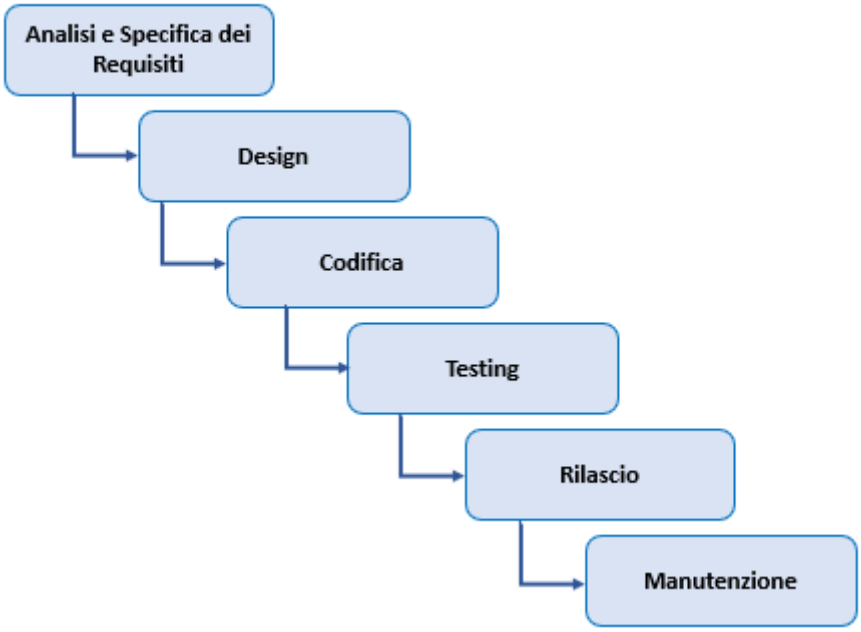


Figure 10 Waterfall Model

1. Analisi e Specifica dei requisiti. I requisiti del prodotto software sono raccolti e definiti dagli sviluppatori. Le attività sono svolte da utenti e da sviluppatori software, gli utenti si fanno portavoce di quelli che sono i bisogni dei consumatori. I bisogni sono un oggetto qualitativo che va convertito in un oggetto quantitativo su cui i progettisti possano lavorare e prendere le decisioni progettuali. Bisogna quindi passare dai bisogni ai requisiti. Il risultato di questa fase è quindi un documento che descriva cosa il sistema debba fare.
2. Design. Durante la fase di progettazione i requisiti sono trasformati in specifiche progettuali, strutturate in modo da poter essere implementate. Viene determinata l'architettura del sistema. Il risultato di questa fase è un documento progettuale che identifica i moduli e le interfacce del sistema.
3. Codifica. I moduli vengono implementati con uno specifico linguaggio di programmazione e successivamente vengono testati. Il risultato è il codice ed il test dei singoli moduli.
4. Integrazione. I moduli vengono fra loro integrati e poi testati. Il sistema è quindi completamente implementato e testato.

In ogni fase si lavora sul prodotto software nella sua interezza. Per questo motivo è necessario che i requisiti siano chiari fin dalle prime fasi progettuali, così come deve esser chiaro il modo in cui si articoleranno le fasi implementative. Da ciò deriva la rigidità del modello. Un errore nei requisiti si ripercuote sull'intera fase progettuale comportando ritardi e rework.[7]

[7] Rework

In un modello a cascata le modifiche del prodotto in corso d'opera sono quindi molto costose poiché richiedono la modifica del prodotto e la ripetizione delle fasi già svolte.

La scarsa flessibilità lo rende poco applicabile a progetti che guardano al lungo periodo e richiedono modifiche e sviluppi successivi. Se i requisiti non sono chiari dal principio, si corre il rischio di consegnare al cliente un prodotto interamente difettoso.

Le valutazioni di migliaia di progetti hanno dimostrato che i difetti introdotti durante i requisiti e la progettazione rappresentano quasi la metà del numero totale di difetti. Inoltre, il costo dell'aggiustare un difetto, aumenta lungo il ciclo di vita di sviluppo. Prima un difetto viene rilevato, meno costoso è il costo della fix.

Vantaggi:

- Il modello è facile da comprendere da implementare
- Le fasi e le attività sono ben definite: aiuta a pianificare e programmare il progetto, consentendo di identificare in anticipo le milestone e i deliverables di ciascuna fase
- L'intensità della pianificazione e della documentazione richiesta, in aggiunta alle revisioni al termine di ciascuna fase, consente di monitorare la qualità del prodotto efficacemente
- La pianificazione consente di monitorare il progresso del processo

Svantaggi:

- I progetti reali raramente si sviluppano in ordine sequenziale e sebbene il modello consenta iterazioni e ripetizioni, queste possono creare disordine e confusione e comportare ulteriori ritardi
- L'effettivo sviluppo avviene tardi nel processo e i risultati del prodotto non sono visibili né verificabili a lungo. Pertanto, è impossibile mostrare e consultare il cliente su una versione funzionante del software se non nelle ultime fasi progettuali
- Si basa sull'assunto che i requisiti di un sistema non possano essere modificati
- È difficile risalire lungo una catena e riprendere una fase dall'inizio una volta che si è terminata
- È poco flessibile: la correzione dei difetti rilevati avviene una volta completate tutte le fasi e pertanto è spesso difficile e costosa
- È difficile integrare attività di risk management
- L'eccessiva richiesta di documentazione porta ad eccessivi costi
- È costoso e richiede tempistiche lunghe

Il modello può esser utilizzato quando:

- I requisiti non variano frequentemente
- Il progetto software non è troppo grande o troppo complesso
- Il contesto è stabile
- Tecnologie e strumenti sono noti e stabili
- Le risorse sono disponibili e competenti
- Le organizzazioni sono molto gerarchizzate e burocratizzate

(Govardhan2, 2010)

Il modello tradizionale a cascata è stato criticato perché comporta una serie di svantaggi come un'alta rischiosità, tempi di consegna lunghi, la necessità di investimenti iniziali ingenti e la sua rigidità. Sebbene il modello a cascata comporti anche dei vantaggi e risulti estremamente utile per grandi progetti, organizzazioni ed esperti si sono mossi verso la ricerca di approcci alternativi al fine di mitigarne le negatività.

2.2.2 Concurrent Engineering per il Software

La Concurrent Engineering consente un rapido sviluppo di un prodotto software attraverso la sovrapposizione di attività. Quando due attività sono legate da una relazione di tipo Finish to Start, ovvero seguono un ordine sequenziale, è possibile ridurre i tempi necessari al processo di sviluppo sovrapponendo le attività. (Ahmed, 2011)

Le fasi dello sviluppo software maggiormente labor-intensive sono le attività di sviluppo e le attività di test. In questo caso, si può implementare un approccio concurrency, dividendo le fasi in sottofasi indipendenti o che comunque possono essere svolte in maniera indipendente e affidando ciascuna delle sottofasi a team di sviluppatori o tester differenti, queste attività possono essere performate in parallelo.

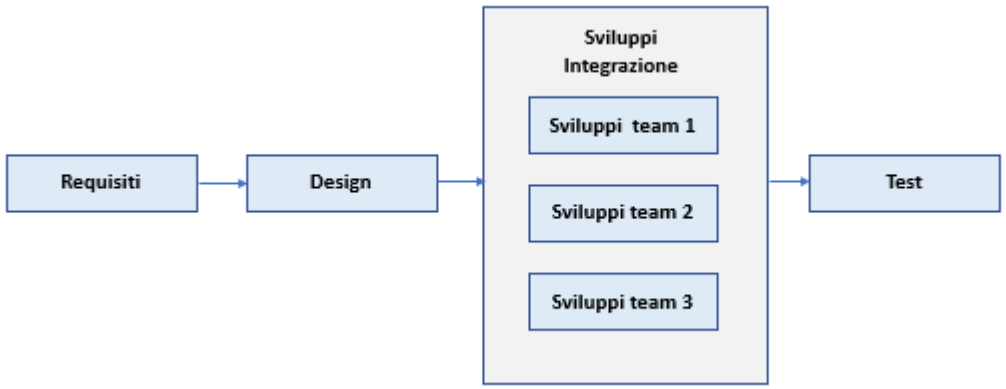


Figure 11 Concurrency applicata alle attività di sviluppo

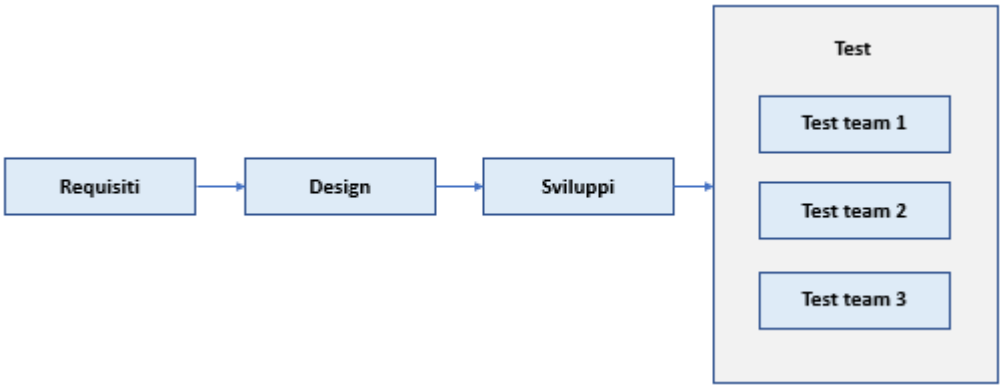


Figure 12 Concurrency applicata alle attività di Testing

Questo meccanismo consente di implementare facilmente la concurrency.

Una volta che gli sviluppi sono portati a termine, le singole componenti sviluppate vengono riunite e successivamente viene verificato che le singole componenti siano in grado di interagire e di lavorare insieme.

Dividere il prodotto software è tuttavia difficile. Durante la fase di progettazione gli sviluppatori decidono come suddividere il prodotto, in modo che molte parti del software possano essere costruite o testate in parallelo: il prodotto software è diviso in modo tale che ciascuna delle parti abbia interfacce definite attraverso le quali successivamente possono essere integrate con le altre.

L'adozione di un approccio concurrency consente di ridurre i tempi di sviluppo ma introduce degli elementi di complessità all'interno del progetto legati alla gestione di sviluppi paralleli e alla sovrapposizione di diverse fasi fra di loro.

Se ad esempio fra la fase di definizione dei requisiti e quella di design la sovrapposizione non è minima, i progettisti dovrebbero prendere le loro decisioni su requisiti non stabili. In tal se cambiassero dei requisiti si genererebbe del rework che oltra ad esser costoso potrebbe portare ritardi nell'esecuzione del progetto.

V. Krishnan e altri suggeriscono che le valutazioni sull'adozione di un approccio concurrency vanno effettuate in base alla sensibilità progettuale, ovvero all'impatto dei cambiamenti upstream sul downstream rework e al grado di incertezza progettuale, ovvero in base a l'incertezza varia durante il processo di sviluppo prodotto. Quando l'incertezza è bassa o comunque diminuisce velocemente durante il processo, il concurrency può essere adottato. (Krishnan, 1997)

2.2.3 Waterfall Modificato

Il Waterfall modificato riprende le fasi del Waterfall tradizionale ma il criterio di transizione fra le fasi ovvero la sequenzialità, è applicato in maniera meno rigida e sono consentite sovrapposizioni fra le fasi. Inoltre, quando è necessario consente di dividere le fasi in sottofasi.

Il Waterfall può essere ulteriormente modificato, introducendo un'iterazione del modello a spirale che consente di valutare i rischi prima di dar inizio al Waterfall vero e proprio. Il modello può esser modificato in vari modi, utilizzando metodi di prototipazione o consentendo la sovrapposizione delle fasi di raccolta e analisi dei requisiti.

Vantaggi:

- Il modello è più flessibile rispetto al Waterfall tradizionale
- Se il team che svolge le fasi è lo stesso, la documentazione può esser notevolmente ridotta
- L'implementazione di facili sviluppi non necessita di attendere che vengano prima svolti quelli difficili.

Svantaggi:

- Le milestone possono risultare maggiormente ambigue rispetto al puro Waterfall
- Le attività svolte in parallelo sono soggette problemi di comunicazione e assunti sbagliati
- Dipendenze non rilevate possono portare a problemi

(Govardhan2, 2010)

2.2.4 V model

Il V model è un'estensione del modello a cascata, dove ad ogni fase nella fase di sviluppo corrisponde una analoga fase di test. Il nome del modello deriva dalla sua modalità di raffigurazione.

Il modello prevede 4 fasi di progettazione, una fase di implementazione e 4 fasi di testing o integrazione. Come per il modello a cascata, la fase successiva non può esser iniziata se non si è completata la fase che la precede.

Il nome deriva dalla forma della sua rappresentazione: il lato sinistro del modello raffigura il Software Development Life Cycle (SDLC), quello sinistro il relativo Software Test Life Cycle (STLC).

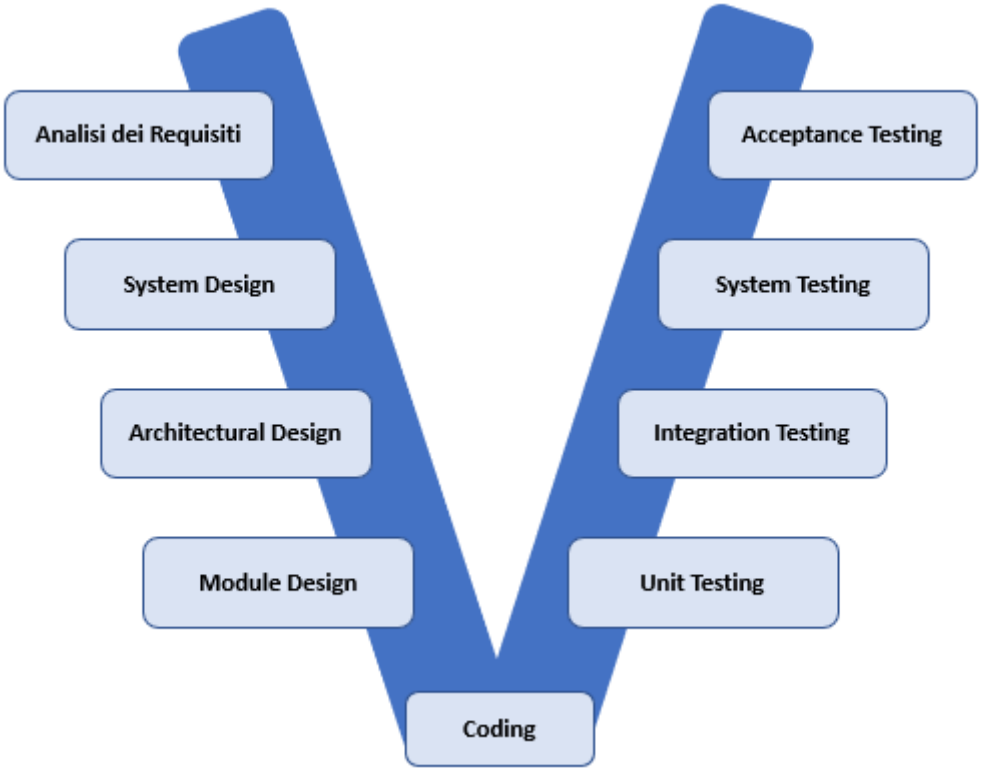


Figure 13 V- Model

Le fasi di progettazione sono:

1. Analisi dei Requisiti. È la fase in cui si raccolgono i requisiti dagli utenti e viene pianificata e progettata la fase di Acceptance Testing. Infatti, i requisiti sono utilizzati per definire i casi di test su cui effettuare tali test.
2. System Design. È la fase in cui viene progettato l'intero sistema e deciso il piano di system test, ciò consente di aver più tempo in seguito per l'esecuzione dei test.
3. Architectural Design. È la fase di High Level Design (HLD) in cui vengono decise e comprese le specifiche architetturali, sulla base di elementi tecnici e finanziari. In questa fase vengono definiti le modalità di comunicazione fra moduli interni e fra l'interno e l'esterno, in base a cui vengono quindi, pianificati i test di integrazione.

4. Module Design. Viene dettagliato il progetto dei moduli interni al sistema Low Level Design (LLD) e decisi i casi di test da eseguire in fase di Unit Test.

Alla progettazione segue la fase di Coding. È la fase di implementazione dei moduli del sistema. Il linguaggio di programmazione è scelto a seconda delle caratteristiche dell'architettura di sistema.

Una volta implementato il sistema, deve esser verificato e validato:

1. Unit Testing. È il test a livello di codice ed è utile ad eliminare i bug[8] nelle prime fasi del processo di sviluppo.
2. Integration Testing. I test di integrazione servono a verificare che i moduli all'interno del sistema siano in grado di comunicare e coesistere al suo interno.
3. System Testing. Il system test è utile per verificare la funzionalità dell'intero sistema e la comunicazione del sistema stesso con l'ambiente esterno. Durante questa fase è possibile scoprire problemi riguardo alla compatibilità fra hardware e software.
4. Acceptance Testing. È il test volto a verificare l'esistenza di problemi funzionali, di performance o di non compatibilità con l'ambiente in cui opera l'utente.

Il V-Model è stato criticato per esser troppo esemplificativo per rispecchiare con fedeltà il processo di sviluppo software e ciò potrebbe portare i manager ad un falso senso di sicurezza nei confronti della progettualità. Come il Waterfall è un modello rigido che non risponde bene ai cambiamenti, a differenza del Waterfall pone evidenza sulle fasi di test, sebbene le ponga in una ridotta finestra temporale al termine dell'implementazione.

È opportuno utilizzarlo nel caso di requisiti e strumenti di progettazione ben noti.

<u>Vantaggi:</u>

- Si tratta di un modello semplice da comprendere e da applicare
- Ogni fase ha risultati specifici ed è seguita da un momento di revisione
- Le attività di test e i casi di test vengono decisi in anticipo, ciò consente di risparmiare tempo
- La verifica e validazione del prodotto avviene nelle prime fasi di sviluppo del prodotto e consente di scoprire presto eventuali difetti

<u>Svantaggi:</u>

- Si tratta di un modello non flessibile, poco adatto a progetti dove c'è rischio di variazioni
- Variazioni in merito ai requisiti risultano molto costose da gestire e richiedono un aggiornamento dei casi di test e non solo delle decisioni progettuali
- Il software viene sviluppato durante la fase di implementazione, quindi non vengono prodotti prototipi del software, per cui vi è il rischio di non soddisfare le aspettative degli utenti

Il modello può esser utilizzato quando:

[8] bug

- I requisiti sono ben definiti, definiti e non ambigui
- Il progetto software non richiedere tempistiche di esecuzione lunghe
- Tecnologie e strumenti sono noti e stabili (Govardhan2, 2010)

2.3 I MODELLI ITERATIVI

I modelli lineari sono difficilmente applicabili nel caso i requisiti non siano ben definiti sin da subito oppure quando i requisiti cambiano durante il processo di sviluppo. In particolare, in contesti innovativi non solo i requisiti variano rapidamente, ma spesso gli utenti non hanno ben chiari quali sono i loro bisogni ed esigenze rispetto ad un'innovazione.

In molti casi gli utenti finali sono in grado di esprimere chiaramente i requisiti solo dopo aver visto un prototipo del progetto. Quindi se il prototipo funzionante è mostrato agli utenti solo al termine delle fasi progettuali come avviene per i modelli lineari, si corre il rischio di consegnare un prodotto che non sia in grado di soddisfare le esigenze dei consumatori e che sia necessario apporre sostanziali modifiche, il che comporta maggiori investimenti in termini economici ma anche di tempo.

Una soluzione è quella di adottare un approccio iterativo, ovvero invece di raccogliere tutti i requisiti fin da subito e sviluppare il software sulla base di questi, si può decidere di raccogliere un set iniziale di requisiti su cui costruire un primo prototipo di software da mostrare agli utenti, una volta completati gli sviluppi. A questo punto gli utenti possono richiedere modifiche che possono essere incorporate velocemente all'interno del prodotto software. Lo svolgimento di queste attività ha una durata variabile di alcune settimane che dipende dal progetto. Terminata questa fase, si prende in considerazione un nuovo e diverso set di requisiti riguardante la progettualità e si ripetono ciclicamente le attività, finché tutti i requisiti non sono convertiti un prodotto software funzionante.

In questo modo il rischio di rilasciare un prodotto che non sia in grado di soddisfare in consumatori è notevolmente ridotto.

Nel corso degli anni sono stati sviluppati molti modelli di sviluppo software che abilitano i processi iterativi e incrementali. I primi modelli sviluppati secondo questa logica sono modelli iterativi a cui sono poi seguiti i modelli basati sulla metodologia agile.

Tuttavia, non tutti i prodotti software possono essere sviluppati secondo approcci iterativi: il modello è adatto a progetti relativamente piccoli. Per i progetti di grandi dimensioni è in effetti ancora preferita l'adozione di modelli lineari, come il modello a cascata che spesso vengono sviluppati simultaneamente da diversi team di progetto.

Tuttavia, i modelli di sviluppo software si stanno sviluppando per poter essere adatti anche a progetti di grandi dimensioni. Per progetti di grandi dimensioni è necessario predisporre un framework stabile, all'interno di questo poi è possibile adottare approcci diversi, anche agili. (Ahmed, 2011)

2.3.1 Il modello a spirale

Il modello a spirale è un modello iterativo caratterizzato dal fatto che i rischi vengono valutati in maniera continua. Il modello fu descritto per la prima volta nel 1986 da Barry Bohem nel suo articolo "A Spiral Model of Software Development and Enhancement".

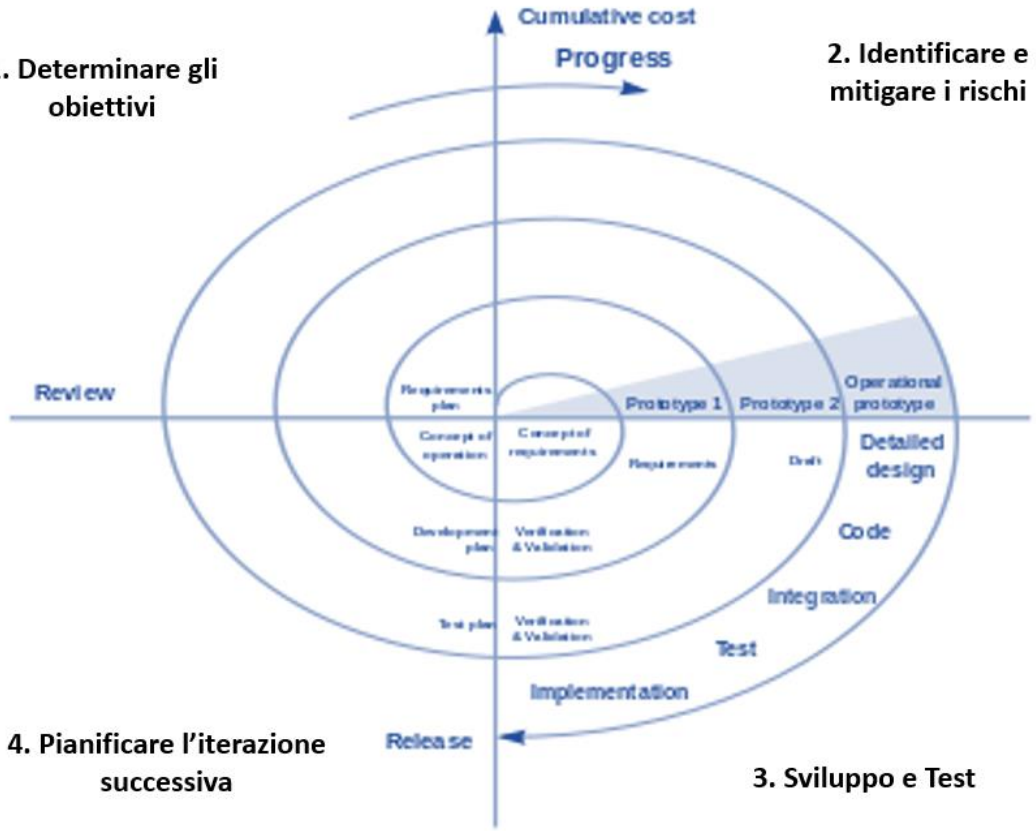


Figure 14 Modello a Spirale

Il progetto di sviluppo software attraversa iterativamente, in una spirale, quattro fasi:

1. Determinazione gli obiettivi. Nella fase iniziale vengono raccolti i bisogni provenienti da tutti gli stakeholder e definiti i requisiti
2. Analisi dei rischi. Nella seconda fase vengono definiti e analizzati i possibili rischi correlati alle soluzioni identificate e vengono successivamente realizzati i primi prototipi
3. Sviluppo e verifica. Il prodotto software viene sviluppato e testato. Il modello di processo di sviluppo può differire in base al tipo di progetto
4. Revisione e pianificazione della iterazione successiva. Il prodotto fino a questo momento sviluppato è verificato e validato e viene pianificata la successiva iterazione

Le fasi sono ripetute fino al completamento, abilitando molteplici cicli di perfezionamento.

Il modello a spirale viene in genere utilizzato per progetti di grandi dimensioni. Consente ai team di sviluppo di creare un prodotto personalizzato e differenziato. Le iterazioni, infatti, abilitano l'aggiunta di nuove funzionalità in fasi successive dello sviluppo. Ad ogni iterazione viene inoltre, realizzata parte del software che può esser mostrato agli utenti e così ricevere i loro feedback sin da subito. Si tratta quindi di un modello adatto a requisiti instabili.

Un altro vantaggio di questo modello è la gestione del rischio. I fattori di rischio infatti, possono variare a seconda di come evolve il progetto e eventualmente, il contesto di riferimento. Il modello analizza i rischi ad ogni iterazione considerando quelli correlati all'evoluzione della progettualità. L'analisi dei rischi richiede comunque il coinvolgimento di esperti del settore e il successo del modello dipende prevalentemente dall'analisi dei rischi. Si tratta di un modello complesso e difficile da implementare. La corretta esecuzione del modello richiede comunque elevati costi.

Vantaggi:

- È un modello flessibile, adatto a requisiti instabili
- Raccoglie i feedback degli utenti sin dalle prime fasi progettuali
- Consente di ridurre dei rischi che vengono analizzati iterativamente
- Abilita l'aggiunta di nuove funzionalità durante lo sviluppo
- Consente di sviluppare rapidamente e ridurre i tempi

Svantaggi:

- Richiede competenze specifiche per l'analisi e la valutazione dei rischi
- Comporta costi elevati e lunghe tempistiche per raggiungere il prodotto finale
- Si tratta di un modello complesso, difficile da implementare correttamente
- Non è adatto a piccoli progetti

Il modello può essere utilizzato quando:

- Il progetto ha un elevato grado di rischio o quando la valutazione e i costi del rischio sono rilevanti per la riuscita del progetto
- Il progetto è un progetto di lungo termine ed i requisiti sono complessi
- Gli utenti non hanno certezza dei loro bisogni
- Sono attesi significativi cambiamenti
- Il progetto è ampio e caratterizzato da frequenti release

(Govardhan2, 2010)

2.3.2 Rational Unified Process

Il Rational Unified Process o RUP è un modello di sviluppo software incrementale che utilizza un approccio iterativo. Il modello è stato sviluppato dalla software house Rational Software, poi acquisita da IBM. (Rational)

Nel 1997 Rational aveva identificato le sei best practise dalla moderna software engineering:

1. Sviluppare in maniera iterativa, tenendo in considerazione i rischi
2. Gestire i requisiti, tenendoli aggiornati e tenendo traccia delle informazioni relative alle relazioni fra requisiti e altre parti del software
3. Utilizzare una architettura basata sui componenti che cooperano e sono indipendenti e allo stesso tempo altamente complementari

4. Realizzare un modello visivo del software, ovvero utilizzare dei diagrammi visuali come gli UML, in modo da rendere gli artefatti più facili da comprendere e trovare più
5. Trovare un accordo fra gli stakeholder
6. Verificare la qualità continuamente che viene abilitato da processi iterativi
7. Gestire e controllare i cambiamenti e le modifiche

Tutto questo è stato il punto di partenza per l'elaborazione del RUP. Con l'introduzione del RUP venne per la rima volta concepito un progetto come un'attività iterativa. Successivamente, sono stati introdotti i modelli agili sulla base dello stesso principio.

In ogni iterazione gli sviluppatori svolgono le seguenti attività:

1. Identificano quale funzionalità verrà sviluppata durante l'iterazione
2. Decidono e progettano il design per il caso d'uso considerato, in base all'architettura prescelta
3. Implementano il design e sviluppano le componenti software
4. Il componente software viene verificato in relazione ai casi d'uso per verificare che ogni componente sia in grado di soddisfarli
5. Rilasciano il prodotto, in seguito alla verifica. Il rilascio può esser rivolto all'intero mercato o solo ad alcuni stakeholder

Ad ogni iterazione vengono ripetute quattro fasi: inception, elaboration, construction e transition.

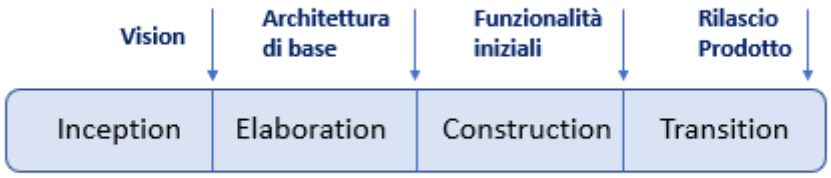


Figure 15 Fasi del RUP

1. **Inception**: è la fase iniziale in cui partendo dall'idea, si giunge alla visione finale del prodotto. Durante questa fase si delimita il project scope e si definisce il business case per il prodotto, ovvero si dettaglia la proposizione di valore del prodotto, quali si considerano fattori di successo ed i principali rischi e quali sono le risorse necessarie per completare il progetto.
   Questa fase consente di definire quali sono i principali destinatari del prodotto ed in che modo questo porterà loro dei benefici. Si definiscono quindi, i principali casi d'uso del prodotto. In base a questi si sviluppa un'architettura del sistema iniziale esemplificativa che indichi i principali sottosistemi. Vengono infine, prodotti una stima iniziale di costi e rischi associati al prodotto ed una bozza di pianificazione del progetto che riporta le principali milestone progettuali. In questa fase possono esser anche prodotti dei prototipi. La fase può dirsi conclusa una volta verificato che sono soddisfatti alcuni criteri:
   - gli stakeholder sono concordi sullo scopo e la definizione del prodotto, ma anche sulle stime di costi e tempi

- i requisiti sono stati correttamente definiti (viene valutato attraverso la bontà dei casi d'uso identificati e dei prototipi se questi vengono sviluppati)
- le stime di costi e tempi, delle priorità definite, nonché del risk assestment e delle eventuali misure di mitigazione identificate sono credibili.

Se i criteri non vengono soddisfatti, il progetto potrebbe esser cancellato o comunque necessita di esser ripensato e rimodulato.

2. **Elaboration.** La seconda fase del RUP prevede uno studio approfondito del contesto di riferimento progettuale, che consente di elaborare dei casi d'uso quasi completi e di introdurre nuovi requisiti, anche non funzionali. Lavorando secondo logiche incrementali, si procede ad una definizione più di dettaglio dell'architettura del sistema e delle stime di tempi e costi. Viene quindi definito il design di basso livello e vengono elaborati i test case. Infine, si procede all'eliminazione dei maggiori rischi associati alla progettualità e realizzato un prototipo che mostra che i principali rischi sono stati mitigati. Al termine della fase vi è un momento di revisione. Secondo criteri prestabiliti, viene presa la decisione di proseguire alla fase successiva oppure se è necessario ripensare o rimodulare alcuni aspetti del progetto.

3. **Construction**. È la fase in cui tutte le caratteristiche del prodotto vengono sviluppate e successivamente testate per verificarne la validità e la conformità rispetto agli standard quantitativi predeterminati. Il prodotto rilasciato viene rilasciato in forma beta, ovvero un rilascio iniziale destinato ad un sottoinsieme selezionato di utenti. Al termine della fase si decide se il prodotto è pronto per esser rilasciato sul mercato oppure il rilascio va ripianificato ad una release successiva.

4. **Transition**. È la fase di rilascio e manutenzione del sistema. In seguito al rilascio sono gli utenti effettivi ad interagire con il sistema e possono riportare bug del sistema oppure la necessità di miglioramenti o di nuove funzionalità. Ma le segnalazioni possono nascere non solo durante l'interazione degli utenti con il sistema ma anche dal fatto, ad esempio, che il sistema deve operare in nuovo contesto. Bisogna dar risposta alle segnalazioni ricevute attraverso azioni di manutenzione, in particolare si parla di:
   - manutenzione correttiva per le segnalazioni di bug nel sistema
   - manutenzione di perfezionamento riguardo alla richiesta di nuove feature
   - manutenzione adattiva per rispondere ai cambiamenti ambientali

   Tutto questo porterà ad un nuovo rilascio del software. In questa fase si eseguono altre attività quali: implementare un servizio di assistenza clienti e di customer care.

I cicli di un processo di sviluppo non sono completamente disgiunti e può succedere che si sovrappongano in alcuni punti, questo può accadere nella fase di transizione, dove si inizia a pensare all'inizio della fase successiva.

Il ciclo di vita di un processo di sviluppo software è costituito dalla ripetizione di una serie di cicli, definiti incrementi. Ogni ciclo include le fasi principali del processo di sviluppo software e termina con il rilascio, interno o esterno, di un prodotto. Ogni fase (inception, elaboration, construction, transition) può a sua volta includere diverse ripetizioni che hanno a che fare, ad esempio, con i

differenti casi d'uso identificati, partendo da quelli caratterizzati dal maggior grado di rischio fino a giungere a quelli caratterizzati da un minor livello di rischiosità.

Se si guarda alle attività che vengono tradizionalmente svolte in un processo di sviluppo software sono distribuite fra le fasi che costituiscono i cicli del RUP.

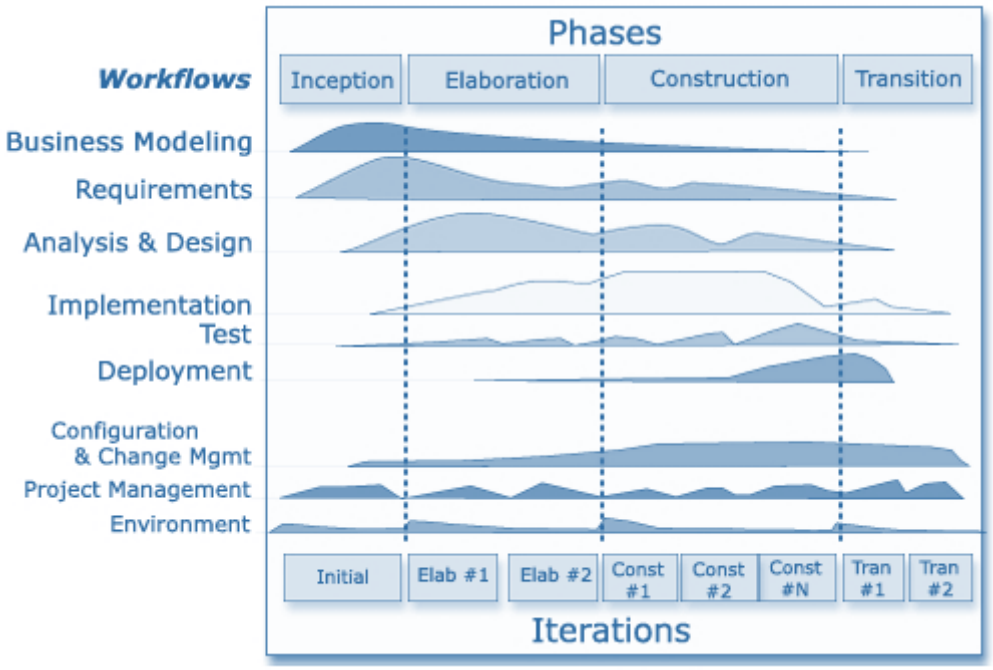


Figure 16 Il RUP e le fasi dei metodi tradizionali

Vantaggi:

- È un modello flessibile che consente al progetto di adattarsi alle richieste di cambiamenti durante l'iterazione delle varie fasi
- Pone l'enfasi sulla necessità di provvedere ad un'accurata documentazione
- Definisce dettagliatamente ruoli, attività, input e output per ogni attività
- Fornisce documentazione di supporto per la sua implementazione
- Abilita lo sviluppo incrementale del prodotto

Svantaggi:

- È un modello complesso da gestire e da porre in atto, specialmente per piccoli team
- Richiede membri del team con elevate competenze tecniche e conoscenze in merito all'applicazione dell'UML
- È costoso e richiede la gestione di una gran quantità di documentazione

Il modello può essere adottato quando:

- Il progetto è ampio e complesso
- Il team è grande e strutturato

- I cambiamenti sono numerosi e frequenti (in assenza di cambiamenti può risultare controproducente l'adozione del RUP)

2.4 I MODELLI AGILI

Fra la metà degli anni '90 ed i primi anni 2000 nel panorama dei progetti in ambito IT, si assiste alla nascita di nuovi modelli per la gestione dello sviluppo software, accumunati dall'adozione di un approccio poco strutturato, adattivo e da uno sviluppo iterativo ed incrementale condotto da team piccoli e polifunzionali. Questi modelli sono l'espressione della metodologia agile.

Nascono per dare risposta ad un mercato caratterizzato da meccanismi sempre più veloci, in termini competitivi, ma anche di tecnologie che a loro volta cambiano con rapidità. In un contesto sempre più veloce, i consumatori non sono in grado di esprimere con chiarezza i loro bisogni, il che porta a requisiti sempre meno chiari e stabili.

Alla base di tutti questi modelli agili vi è la convinzione che sviluppo del software sia un'attività complessa che può esser semplificata utilizzando le iterazioni.

La prima iterazione riguarderà task e requisiti chiari e ben definiti, requisiti più complessi verranno trattati in seguito, dopo averne migliorato la comprensione. I metodi agili nascono come metodi flessibili per adattarsi ai rapidi cambiamenti dei nuovi contesti di innovazione. (What is Agile?, n.d.)

L'Agile è intesa come l'abilità di creare e rispondere al cambiamento. È un modo di affrontare ed infine aver successo in un ambiente caratterizzato da incertezza e turbolenza.

Sono stati gli autori del Manifesto Agile a scegliere il termine "Agile" in quanto rappresentante dall'adattabilità dei modelli proposti. Il Manifesto Agile fu pubblicato nel 2001, firmato da 17 progettisti, esperti informatici che avevano partecipato alla stesura. Il Manifesto recita:

"Stiamo scoprendo modi migliori di costruire il software realizzandolo e aiutando altri a realizzarlo.
Attribuiamo valore:

A individui e interazioni più che a processi e strumenti

Ad un software che funziona più che ad una documentazione completa

Alla collaborazione col cliente più che alla negoziazione contrattuale

A reagire al cambiamento più che a seguire un piano

I valori a destra sono importanti, ma noi preferiamo quelli a sinistra"

(Agile Manifesto, n.d.)

Alla base della Metodologia Agile e dei valori riportati nel Manifesto ci sono i 12 Principi dell'Agile, riportati di seguito.

1. La nostra priorità è soddisfare il cliente mediante consegne anticipate e continue di software di valore.
2. Le persone dell'azienda e gli sviluppatori devono quotidianamente lavorare assieme durante tutto il progetto
3. Le modifiche ai requisiti sono benvenute, anche nelle ultime fasi dello sviluppo
4. Consegnare di frequente software funzionante
5. Il software funzionante è la prima misura di progresso
6. I progetti si costruiscono attorno a individui motivati. Dategli l'ambiente ed il supporto di cui hanno bisogno, e confidate che facciano il loro lavoro
7. Le migliori architetture, requisiti, e design emergono da team auto-organizzanti
8. Il metodo più efficace ed efficiente di trasmettere informazioni verso e all'interno di uno sviluppo è mediante conversazioni faccia a faccia
9. I processi agili promuovono lo sviluppo sostenibile
10. L'attenzione costante all'eccellenza tecnica e al buon design esalta l'agilità
11. La semplicità è essenziale
12. A intervalli regolari, il team riflette su come diventare più efficace, quindi sintonizza e regola di conseguenza il suo comportamento.

(12 Principles Behind the Agile Manifesto)

2.4.1 Il metodo Agile

Il metodo Agile propone un approccio iterativo ed incrementale allo sviluppo di un prodotto software. Il processo di sviluppo software è suddiviso in modelli su cui i progettisti lavorano.

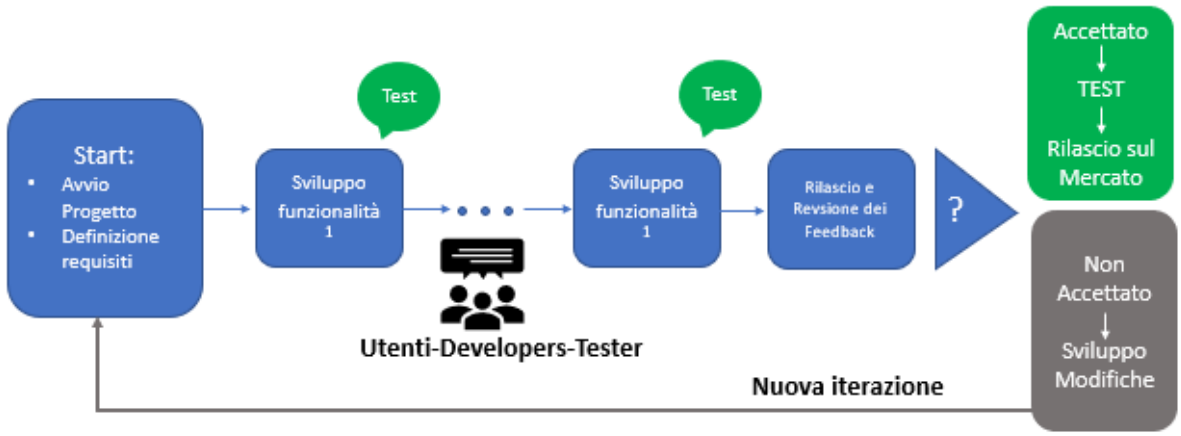


Figure 17 Il Modello Agile

Una delle cose che differenzia l'agile dagli altri approcci allo sviluppo software è il focalizzarsi sulle persone che lavorano e su come queste siano in grado di lavorano assieme. È in effetti basato su

team piccoli e poli funzionali, in grado di organizzare da sé il proprio lavoro. I team hanno l'abilità di comprendere quale approccio utilizzare per la risoluzione di un problema, Non ci sono ruoli ben definiti nel team, ma bisogna assicurarsi che all'interno del team ci siano le skill adatte a risolvere il problema. Ad assicurarsene sono i manager, i quali hanno il compito di creare l'ambiente adatto affinché i team abbiano successo.

Il progetto è svolto attraverso brevi iterazioni (2-4 settimane). La pianificazione è pertanto poco dettagliata. Al termine di ogni iterazione il prodotto o alcune sue funzionalità rilasciabili e osservabili, vengono mostrate agli utenti che hanno quindi la possibilità di verificare il prodotto. L'utente quindi viene coinvolto sin da subito nelle fasi progettuali e ha molteplici possibilità di prender parte alle decisioni relative al progetto, di testare il prodotto e suggerire modifiche e cambiamenti. Al termine di ogni iterazione, l'utente valida e verifica il prodotto, si esegue la *user acceptance*. Il metodo agile richiede che utenti e sviluppatori comunichino frequentemente e collaborino, analizzando i requisiti e pianificando i nuovi. I requisiti vengono infatti trattati in maniera differente rispetto a quanto avviene nei metodi tradizionali: la definizione dei requisiti nei processi agili è infatti anche essa iterativa ed incrementale.

Il successo di un progetto è ottenuto quando il prodotto è consegnato in tempi brevi, evitando gli sprechi e quando è in grado di soddisfare le esigenze degli utenti. Per garantire la corrispondenza ai bisogni dei consumatori, l'utente è coinvolto continuamente nel processo.

Ogni iterazione ha la propria fase di test, ad ogni nuovo rilascio viene eseguito un test di regressione. Così come progettisti e utenti, tester e gli sviluppatori lavorano assieme. La modalità di esecuzione dei test consente di individuare gli errori tempestivamente e di procedere sin da subito alla loro risoluzione.

L'approccio agile è inoltre meno strutturato e document driven rispetto agli approcci tradizionali. Questo consente di ottenere un approccio più flessibile in grado di adeguarsi ai cambiamenti.

I metodi agile abilitano la gestione dei cambiamenti attraverso le iterazioni e consentono di sviluppare dei prodotti software in modo incrementale.

L'approccio consente di mitigare i rischi progettuali e di adattarsi a contesti mutevoli, soddisfacendo i requisiti derivanti dalle real time market research. Durante le attività di sviluppo, si procede in parallelo a ricerche di mercato, attraverso interviste o questionari, per comprendere meglio le esigenze dei clienti che nei contesti di innovazione moderna cambiano frequentemente. In base ai risultati delle ricerche si decide come sviluppare il prodotto finale e quali funzionalità e caratteristiche introdurre. Questo approccio consente di mitigare i rischi legati allo sviluppo di un prodotto che non risponde alle esigenze dei consumatori e che può avere caratteristiche non desiderate e comunque costose.

Vantaggi:

- È un modello flessibile che consente al progetto di adattarsi alle richieste di cambiamenti e alle variazioni delle circostanze
- Pone l'attenzione sulle qualità tecniche del prodotto

- Coinvolge gli utenti nel processo, consentendo di realizzare prodotti rispondenti alle loro esigenze
- Consente di mostrare sin da subito un prodotto agli utenti, grazie ai frequenti rilasci e di testarlo e validarlo (gli errori vengono scoperti presto e diminuisce il costo di risoluzione dell'errore)
- Migliora le comunicazioni fra sviluppatori, tester e utenti
- Abilita lo sviluppo iterativo e incrementale

Svantaggi:

- Richiede che i team di progetto siano relativamente piccoli, il metodo agile infatti pone l'enfasi sulla comunicazione e richiede che questa avvenga non sotto forma scritta ma face-to-face, questo porta alla necessità di dover staffare dei team di progetto ridotti
- Non consente che i team che lavorano al progetto siano dislocati
- Pone poca enfasi sulla necessità di documentazione: una buona documentazione fornisce un mezzo per controllare un progetto. Se manca la documentazione, è difficile controllare un progetto. Ciò significa che se si verifica un errore, è difficile rintracciare la causa principale del problema.
- Coinvolgimento di terze parti: una terza parte non può mai essere coinvolta in un progetto se non c'è abbastanza documentazione inoltre in progetti agili, il lavoro è svolto da un team piccolo e integrato. Ciò rende impossibile affidare una parte del lavoro del progetto a un fornitore di servizi di terze parti.
- In caso di progetti ampi è difficile definire l'effort richiesto sin da subito
- Richiede sviluppatori con esperienza in grado di gestire i cambiamenti
- In caso di requisiti iniziali poco chiari non si ha certezza del risultato finale

Il modello può essere usato quando:

- Il progetto è piccolo
- Sono richiesti molte modifiche e cambiamenti
- I requisiti non sono definiti
- Il prodotto è adatto ad un'architettura flessibile, che possa sostenere i cambiamenti
- L'utente può esser coinvolto nelle fasi di sviluppo

(Govardhan2, 2010)

2.4.2 Il metodo Scrum

Il metodo Scrum è un framework iterativo ed incrementale che si fonda sui principi delle Metodologia Agile. Il metodo fu presentato per la prima volta nel 1995 da Ken Schwaber e Jeff Sutherland e nel 2001 Schwaber insieme a Mike Beedle pubblico il libro *Agile Software Development with Scrum.*

Il metodo Scrum riconosce la possibilità che le circostanze possano cambiare, così come le esigenze degli utenti o che queste possano essere non chiare, per cui un approccio pianificato non è adatto alla gestione dei progetti di sviluppo software.

Lo Scrum si basa sulla teoria dell'empirismo, secondo cui la conoscenza deriva dall'esperienze ed è su questa che si basano le decisioni. Alla base della teoria stanno tre concetti fondamentali:

- Trasparenza: la trasparenza riguardo a processi e decisioni è fondamentale per una comunicazione efficace che porti da una conoscenza comune a tutti.
- Ispezioni: è necessario condurre un controllo riguardo al processo stesso
- Adattamento: che deve esser condotto in maniera efficiente ed in tempi minimi affinché il processo conduca al risultato

Lo sviluppo del prodotto software che segue l'approccio Scrum è suddiviso in iterazioni, note come Sprint, di una durata variabile da una a quattro settimane, decisa dal team. Ogni iterazione ha l'obiettivo di sviluppare delle caratteristiche potenzialmente rilasciabili del prodotto.

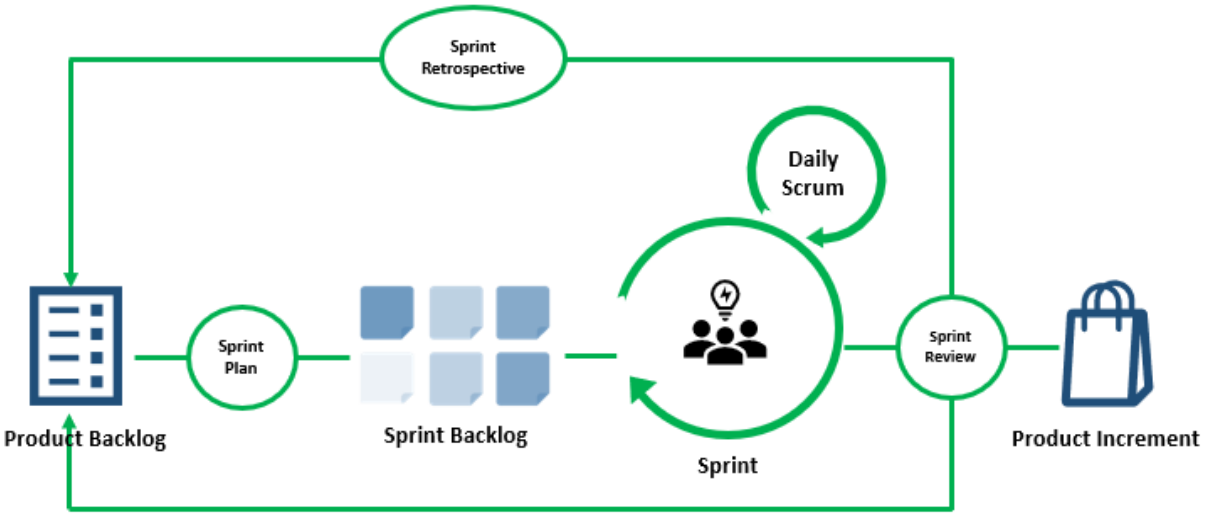


Figure 18 Il modello Scrum

Gli attori principali del metodo Scrum sono tre e costituiscono lo Scrum Team, ovvero un team cross funzionale e auto organizzato, responsabile del rilascio iterativo ed incrementale del prodotto.

- **Il Product Owner**. È il rappresentante delle esigenze degli stakeholder ed il responsabile principale del Product Backlog. Il Product Owner ha il compito di garantire che gli elementi che costituiscono il Product Backlog siano chiari e ordinati a seconda degli obiettivi, che siano compresi dal Team di sviluppo e che sia comprensibile su quale di questi si lavorerà in seguito.
- **Il team di sviluppo**. È un team autorganizzato e polifunzionale, i suoi componenti vengono chiamati sviluppatori. Ha il compito di trasformare gli elementi del Product Backlog in un insieme di caratteristiche del prodotto software, non necessariamente rilasciabili, ad ogni iterazione. Solitamente è composto dalle 4 alle 9 persone.
- **Lo Scrum Master**. È il responsabile del processo Scrum. Ha il compito di rimuovere gli ostacoli al progresso, facilitare l'esecuzione e promuovere i flussi di comunicazione. È di supporto al Product Owner nella gestione del Product Backlog e aiuta lo Scrum Team nella definizione degli elementi del Product Backlog.

Il Product Backlog è la fonte dei requisiti del processo, elencati in ordine di priorità. La lista può esser modificata sia dal Product Owner ma anche dai membri del team che possono aggiungere, rimuovere o modificare i requisiti. Ad ogni iterazione si decide quali dei requisiti andranno sviluppati nella iterazione successiva. Durante lo sprint planning viene infatti definito l'incremento o lo sprint successivo e vengono selezionate le caratteristiche della successiva iterazione in base a: valore, necessità e priorità. I requisiti vengono tradotti in stime e attività da compiere che vanno a costituire lo sprint backlog.

Il processo di sviluppo prodotti si articola quindi in Sprint.

Ogni Sprint inizia con uno Sprint Planning durante il quale lo Scrum Team si accorda sull'obiettivo dello sprint successivo, lo sprint goal, fornendo una descrizione di quale sarà l'output dello sprint. Si selezionano gli elementi del Product Backlog che verranno sviluppati e si predispone lo Sprint Backlog, ovvero una sorta di pianificazione del lavoro da svolgere.

All'interno di ogni Sprint vengono svolti giornalmente degli incontri della durata di 15 minuti, i Daily Scrum. Un Daily Scrum è formato da un momento iniziale utile al team per sincronizzarsi. Vengono valutati i traguardi raggiunti rispetto allo scorso Daily Scrum, viene poi stilata una lista delle cose da fare in vista del prossimo meeting. È il team a condurre l'incontro, lo Scrum Master vigila sul rispetto delle tempistiche.

Al termine dello sprint, viene effettuata una revisione di quanto prodotto: lo Sprint Review. Si discute di quanto fatto, delle cose ancora da fare e delle eventuali migliorie da introdurre ed apportare, si analizza il Product Backlog cercando di fare una previsione sulla data di fine progetto. In seguito, viene condotta la Sprint Retrospective in cui si discute di possibili miglioramenti da apportare al processo, a persone e strumenti per le successive iterazioni, in base a quanto accaduto nello Sprint precedente.

(Sutherland, 2017)

2.4.3 Il metodo Lean

Il Lean Software Development è un metodo di sviluppo software che rientra al di sotto del termine ombrello della Metodologia Agile.

Il metodo Lean si basa su 7 principi:

1. Eliminare gli sprechi. Esempi di sprechi possono essere: difetti nel software, caratteristiche o funzionalità aggiuntive che non portano valore ai consumatori. Essenziale per eliminare gli sprechi è la loro identificazione.
2. Amplificare l'apprendimento. Il processo di sviluppo software è un processo continuo di apprendimento e l'apprendimento può essere incentivato attraverso il continuo dialogo e confronto con gli utenti, che consente una migliore comprensione del contesto e il raggiungimento della soluzione più adatta, oppure attraverso l'iterazione di cicli di sviluppo a cui seguono test di integrazione.

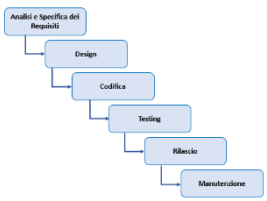
3. Decidere il più tardi possibile. Rimandare alcune decisioni progettuali per fronteggiare ambienti caratterizzati da incertezza, oppure in caso gli utenti non abbiano chiari fin da subito i requisiti.
4. Rilasciare il prima possibile. Il rilascio anticipato consente di ottenere prima i feedback e di migliorare per primi il proprio prodotto.
5. Dare rilevanza al team
6. Costruire integrità. Il cliente deve avere un'esperienza completa del prodotto
7. Ottimizzare il tutto. Verificare che il prodotto sia ottimizzato globalmente e che le performance ottime non siano raggiunte esclusivamente a livello del singolo modulo, ma anche che l'interazione fra i moduli sia ottimizzata.
   (Mary Poppendieck, 2003)

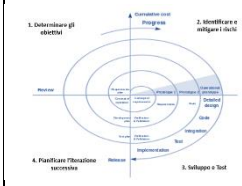
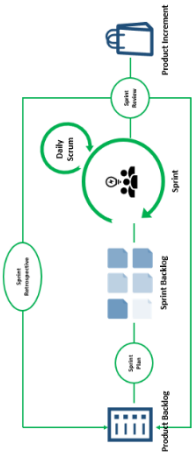
Espressione del metodo Lean è il minimum viable product o MVP. Un MVP è la versione "minima" di un prodotto ovvero quella che possiede le sole caratteristiche necessarie per soddisfare i primi consumatori. Questo tipo di rilascio consente di raccogliere feedback sul prodotto dal mercato e sulla base di questi, apportare miglioramenti al prodotto o inserire nuove funzionalità in seguito. Questa pratica diminuisce i rischi e i costi del rilascio di un nuovo prodotto. Infatti, se si rilasciasse un prodotto completo di funzionalità, sarebbero richiesti tempi più lunghi e alcune delle funzionalità potrebbero risultare non utili o non conformi alle aspettative degli utenti. (Bonzer Technology, 2016)

Il concetto di MVP fu promosso da Eric Ries nel suo libro Lean Startup.

2.5 UN APPROCCIO COMPARATIVO PER LA SCELTA DEL MODELLO DI GESTIONE DELLO SVILUPPO SOFTWARE

Table 1 Riepilogo Caratteristiche dei Software Development Method

Modello		Vantaggi	Svantaggi	Applicabilità
Waterfall (Modelli lineari)		Facile da implementare	Costoso	I requisiti e il contesto sono stabili il progetto software non è troppo grande o troppo complesso, tecnologie e strumenti sono noti le risorse sono disponibili e competenti, in organizzazioni molto gerarchizzate e burocratizzate
		Fasi e attività ben definite	Richiede tempistiche lunghe	
		Controllo della qualità di processo e prodotto	Rigido: non gestisce i cambiamenti	
		Monitoraggio	Richiede che i requisiti siano stabili	
		Documentazione accurata per tracciamento attività	Le iterazioni di attività generano confusione e ritardi	
Iterativi Incrementali		Modello flessibile, adatto a requisiti instabili	Non è adatto a piccoli progetti	i requisiti sono complessi o incerti il progetto è un progetto di lungo termine il team è strutturato il progetto ha un elevato grado di rischio
		Raccoglie i feedback degli utenti anticipatamente	Modello complesso, difficile da implementare	

		Consente di sviluppare rapidamente e ridurre i tempi	Comporta costi elevati e lunghe tempistiche	il progetto è ampio e caratterizzato da frequenti release
		Abilita l'aggiunta di nuove funzionalità durante lo sviluppo	Richiede competenze specifiche per gestire i rischi o per l'adozione di specifici linguaggi (UML)	
		Gestisce il rischio (spirale)		
Agile		Modello flessibile si adatta ai cambiamenti	Team di progetto piccoli, riuniti in un'unica location	Il progetto è piccolo Sono richiesti molte modifiche e cambiamenti I requisiti non sono definiti Il prodotto è adatto ad un'architettura flessibile L'utente può esser coinvolto nelle fasi di sviluppo
		Errori scoperti nelle prime fasi	Poca documentazione e pianificazione, basso monitoraggio	
		Sviluppo iterativo e incrementale	Non consente di affidare parte del lavoro a fornitori	
		Ampio coinvolgimento utenti nel processo	In caso di progetti ampi è difficile definire l'effort richiesto sin da subito	
		Validazione continua da parte degli utenti	Necessari sviluppatori con esperienza	
		Migliora le comunicazioni fra sviluppatori, tester e utenti	In caso di requisiti iniziali poco chiari non si ha certezza del risultato finale	

Se si guarda alla tabella in alto è evidente che non esiste un modello che possa essere applicato per tutte le tipologie di progetto. Ma la scelta del modello da adottare è una determinante per il successo di un progetto.

Alcuni metodi di processo possono adattarsi a un particolare tipo di progetto, altri no e possono rivelarsi più adatti per progetti differenti.

I metodi agili e i modelli iterativi sono utili nel caso sia necessario sviluppare un prodotto software velocemente, il progetto non sia particolarmente ampio e possano essere utilizzati team piccoli che consenta l'adozione di metodologie agili. Se il progetto è ampio infatti, il ridotto numero di sviluppatori che abilita i team agili comporterebbe tempistiche lunghe.
Nel caso di progetti più ampi e complessi che richiedono un numero di sviluppatori maggiore di 50 possono essere utilizzati modelli iterativi, come il Rational Unified Process.

Oppure in caso di un progetto ampio e complesso e necessità di un delivery rapido può essere utilizzato un approccio di tipo concurrent.

Nel caso i requisiti progettuali non siano chiari fin da subito, può esser difficile valutare la grandezza del progetto ed è necessario un approccio cautelativo, abilitato da metodi agili.

Se invece il progetto riguarda un prodotto noto ma ci si pone come obiettivo quello di migliorare in termini di performance di processo, di aumentare la produttività oppure la qualità, gli approcci tradizionali plan drive hanno la meglio sui metodi agili.

Per usufruire dei vantaggi di entrambi i metodi si potrebbe utilizzare un approccio ibrido, introducendo nel modello a cascata driven plan delle logiche agili legati alla gestione di alcune fasi

o l'introduzione di iterazioni fra di esse, o viceversa inserendo componenti della pianificazione all'interno dei modelli agili. Tale approccio non è raro nei casi reali, in seguito se ne vedrà un'applicazione. (Ahmed, 2011)

Alistair Cockburn, nel suo articolo: "Selecting a Project's Methodology" afferma che utilizzare differenti metodologie è appropriato e necessario. Cockburn in seguito alle sue analisi ha sviluppato un framework per la selezione della metodologia da adottare. Il suo approccio si basa su 4 principi che egli deriva dalle sue ricerche e su due fattori, che egli identifica come driver principali che guidano la scelta metodologica. (Cockburn, 2000)

I 4 principi:

1. Team più ampi richiedono una metodologia più ampia (cioè una metodologia che comprensiva di più elementi, ruoli, standard, linguaggi)
2. I sistemi più critici (ovvero i sistemi i cui difetti comportano maggiori più gravi) necessitano di una maggiore visibile correttezza nel loro sviluppo. Ovvero per progetti i cui difetti potrebbero causare perdite in termini di vite umane, si investirà maggiormente nella qualità dei loro strumenti rispetto al progetto di un e-commerce.
3. Esiste una relazione fra: numerosità del team, tipo di metodologia e dimensione del problema. Se un team piccolo è in grado di portare a termine il progetto, allora si utilizzerà una metodologia più leggera (ad esempio una metodologia agile), se il problema diventa più ampio sarà necessario un team ampio per rilasciare nei tempi previsti e conseguentemente una metodologia più "pesante", ovvero più strutturata.

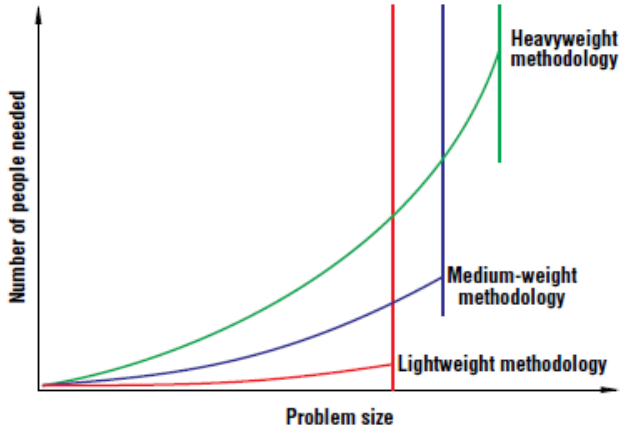


Figure 19 La metodologia e la dimensione del problema impattano sulla numerosità dello staff

4. La forma di comunicazione più efficace per la condivisione di idee) è quella face to face

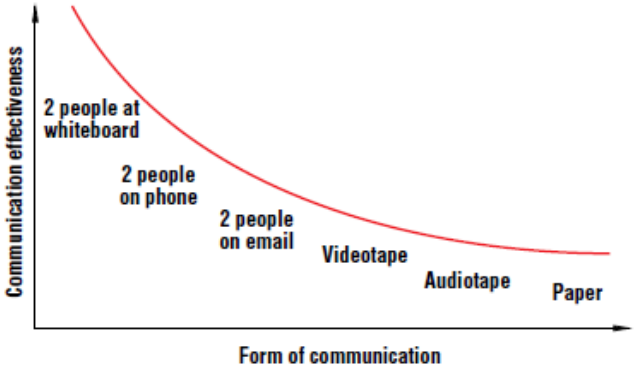


Figure 20 Efficacia delle varie forme di Comunicazione

Altri due fattori determinano quale sia il metodo appropriato:

- le priorità progettuali, se è necessario che il software venga consegnato presto che piuttosto sia privo di difetti o che i costi siano contenuti. Le priorità progettuali definite dagli stakeholder possono essere diverse
- l'esperienza dei progettisti. Le esperienze passate dei progettisti sono una forte determinate della scelta del metodo da utilizzare.

Cockburn ha sviluppato un framework tridimensionale per la scelta del modello con dimensioni: Criticità, Numero di persone dei team, Priorità.

Diversi sono i metodi e i framework presenti in letteratura che si vogliono proporre come strumento di supporto alla decisione. Negli studi più recenti sembra comunque prevalere l'idea di avvalersi di un approccio misto fra metodi tradizionali e metodi agili che gestisca le varie fasi progettuali in base alle loro peculiarità, adottando il metodo ad esse più appropriato.

In particolare, Apoorva e Deepty nel 2013 attraverso uno studio comparativo, concludono che un modello ibrido, opportunamente modificato e appropriato possono essere utili per migliorare la qualità del software. (Deepty, 2013)

3 IL CASO DI STUDIO

L'oggetto del Caso di Studio è un progetto di sviluppo di un prodotto software di uno dei maggiori gruppi bancari italiani. Il progetto nasce e si sviluppa nella sfera della PSD2.

La raccolta dei dati e delle informazioni di supporto all'analisi è avvenuta durante le attività di tirocinio svolte presso la società di consulenza Accenture. Il tirocinio si è svolto presso la sede del cliente, Intesa San Paolo, da Luglio 2019 a Gennaio 2019, le attività condotte sono state di supporto al PMO e al Defect Manager. Durante lo svolgimento delle attività di tirocinio si è avuto modo di osservare e prendere parte ai processi decisionali in merito alle modalità di gestione di un progetto di sviluppo software in un contesto innovativo, caratterizzato da incertezza e turbolenza.

Nel Gennaio 2019 la banca si è posta l'obiettivo di sfruttare una delle opportunità offerte dalla normativa e compiere i primi passi per implementare operativamente una strategia di innovazione presentandosi come innovatore nel panorama dell'Open Banking.

Lo scopo del progetto è quindi il rilascio di una nuova funzionalità gratuita per i clienti della Banca, sul canale Internet Banking e sul canale App Mobile. La funzionalità in oggetto è rivolta a tutti i clienti persone fisiche maggiorenni.

Attraverso l'offerta di una funzionalità aggiuntiva sui propri servizi digitali, la Banca vuole incentivare ed incrementare l'utilizzo dei canali Internet Banking e App Mobile con lo scopo di imparare a conoscere i comportamenti finanziari e digitali dei propri clienti.

In un contesto di competizione crescente, dove il consumatore assume un ruolo sempre più da protagonista è fondamentale infatti, investire per conoscere e definire meglio esigenze e profilo della clientela.

La nuova funzionalità è ottenuta integrando i servizi sviluppati da un protagonista del mondo PSD2: una Fintech.

3.1 IL METODO DI GESTIONE UTILIZZATO E L'EVOLUZIONE DELLA PIANIFICAZIONE

L'orizzonte temporale all'interno del quale avviene il processo di sviluppo prodotto, dalla fase di definizione dei requisiti fino al rilascio in produzione, va da Gennaio 2019 a Gennaio 2020, data in cui avviene il rilascio della funzionalità in modalità MVP (Minimum Viable Product).

La scelta di una prima release in ottica MVP è dettata da due fattori:

- un contesto PSD2 non ancora maturo. La poca maturità del contesto PSD2 genera incertezza e la modalità MVP consente di rimandare le decisioni progettuali in merito a caratteristiche e funzionalità non chiaramente definibili sin dal principio. Le decisioni progettuali vengono poi prese sulla base ai feedback ricevuti dagli utenti, che consentono di minimizzare l'incertezza relativa ad esse. Inoltre, il contesto PSD2 è un contesto in cui realtà diverse comunicano fra loro, la comunicazione è possibile quando ciascuna delle parti ha apportato i cambiamenti necessari per rendere possibile la comunicazione stessa. In un contesto poco

maturo, dove non tutti gli attori sono abilitati agli scambi comunicativi, il rilascio di caratteristiche e funzionalità che si basano su queste, può risultare controproducente ed eventualmente portare a malfunzionamenti

- la necessità di garantire time to market brevi. Queste vengono abilitate dalla progettazione e dallo sviluppo di un prodotto con funzionalità minime. È stato quindi necessario disegnare una ridotta finestra temporale disponibile per gli sviluppi. Le aree strategiche della banca hanno infatti espresso la necessità di una release il più possibile time to market rispetto alla milestone della PSD2 del 14 Settembre, così che la banca possa posizionarsi sul mercato contestualmente ai vari peer.

Due sono quindi i vincoli principali del progetto:

- la necessità di time to market brevi per rimanere competitivi
- la poca maturità del contesto che porta ad incertezza dei requisiti

Al fine di ridurre i rischi progettuali, si è reso necessario definire il perimetro funzionale in ottica MVP e design-to-cost.

Il progetto prevedeva inizialmente di ottenere la nuova funzionalità offerta, integrando i servizi sviluppati da una Fintech. Le attività di integrazione sono condotte da team esperti. I team quindi, hanno una conoscenza approfondita dei processi di integrazione, sebbene ogni integrazione presenti delle peculiarità relative alla specifica funzionalità da integrare.

Il processo di sviluppo prodotto coinvolge differenti team, relativamente piccoli dalle 4 alle 8 persone al massimo, ciascuno con differenti competenze e responsabilità. I team appartengono alla sezione dei Sistemi informativi della Banca e a 5 differenti fornitori, compresa la Fintech, e ciascun team è dislocato in diverse città del Nord Italia.

Il progetto nasce e si sviluppa in un contesto strutturato e gerarchizzato, in cui è richiesta una documentazione accurata delle attività svolte.

Se nella definizione del prodotto oggetto di rilascio si segue un approccio agile, la struttura dei team di sviluppo, la necessità di documentazione e le competenze in merito ai processi di integrazione noti, conducono all'adozione di un approccio più tradizionale per la gestione del processo di integrazione: il V model.

La variazione di uno degli elementi determinanti della progettualità ha introdotto ulteriori elementi di complessità nel progetto e richiesto una modifica della pianificazione e dell'approccio gestionale e metodologico utilizzato. Con l'introdursi di complessità l'approccio tradizionale si evolve verso logiche più agili.

3.1.1 Il Demand Management nella Banca

Il contesto progettuale è un contesto strutturato, dove si seguono precise linee guida di Demand Management, ovvero del metodo di pianificazione utilizzato per la gestione della domanda di nuovi prodotti all'interno delle imprese. Si tratta di un metodo condiviso fra le sezioni del Business e dei

Servizi Informativi della Banca in base a cui queste possano confrontarsi sul soddisfacimento delle aspettative rispetto a quanto prodotto.

Lo scopo dei protocolli del Demand Management è quello di migliorare la qualità dei prodotti e servizi IT, minimizzando i costi, ottimizzando la gestione delle risorse e allineando i progetti con la strategia. Per raggiungere gli obiettivi individuati si utilizzano dei KPI che consentono di misurare il raggiungimento degli obiettivi. La misurazione è abilitata da una documentazione adeguata e dal rispetto dei protocolli.

Le logiche di demand management sono adottate nell'ottica di raggiungimento dell'efficienza organizzativa e di un appropriato monitoraggio nei progetti IT verificandone al contempo la coerenza con le strategie aziendali.

La documentazione deve fornire una chiara rappresentazione dei processi operativi, degli attori e delle relative responsabilità.

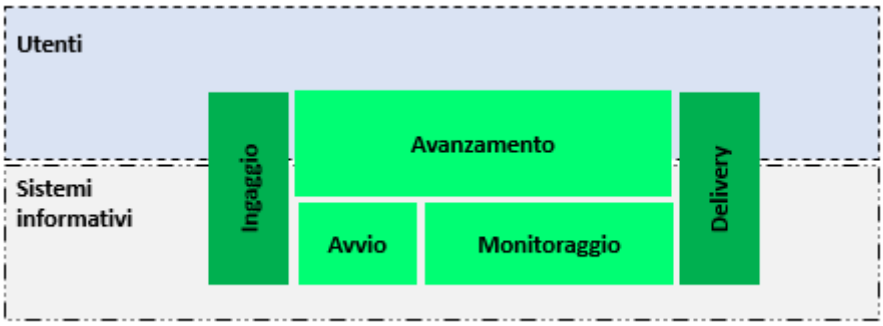


Figure 21 Processo di Demand Management

Il protocollo previsto dalla Banca individua 5 fasi del processo di gestione dello sviluppo di nuovi prodotti:

1. **Ingaggio**. Il responsabile della fase di ingaggio è il Service Manager. Durante questa fase viene definita e formalizzata la richiesta per la nuova progettualità, viene scelto il Service Manager, si tiene il kick off di progetto e a seguito delle prime stime economiche, viene stanziato un budget per il progetto. Analizzando le esigenze e le necessità dell'utente si definisce il contenuto del BRB, ovvero del Business Requirement Book. Il BRB è costituito da due sezioni: la prima, compilata dal Business con il supporto del Service Manager. definisce il contesto di riferimento e gli obiettivi progettuali, la seconda contiene i requisiti tecnici e funzionali. Una volta acquisiti i requisiti, viene definito il piano preliminare, viene stilato l'elenco degli attori coinvolti nella progettualità e consolidate le stime dei costi. La richiesta, formata dal BRB, dalle stime dei costi e dal piano viene valutata dalla Governance in termini di costi e benefici ed eventualmente approvata
2. **Avvio.** Approvata la richiesta si procede alla pianificazione: in termini di tempi e costi e all'ingaggio dei fornitori
3. **Monitoraggio.** Dato l'inizio ai lavori si effettuano azioni di monitoraggio in termini di costi e tempi. In questa fase assume un ruolo fondamentale la documentazione e la condivisione delle informazioni.

4. **Fase di Avanzamento.** Comprende le attività attraverso cui viene mostrato l'avanzamento dello stato dei lavori agli Utenti. È fondamentale in questo caso disporre di strumenti appositi oppure favorire momenti di incontro per la condivisione delle informazioni.
5. **Delivery.** Il progetto si conclude con il rilascio, l'implementazione delle attività di supporto alla produzione e l'accettazione da parte degli Utenti. Anche questa fase deve essere accuratamente documentata.

È fondamentale, in ogni fase della progettualità la condivisione delle informazioni fra le aree informatiche e quelle di business per migliorare l'efficienza e l'efficacia dei processi. La Banca indica l'utilizzo di documentazione a supporto della condivisione di informazioni. (Intesa San Paolo)

3.1.2 Pianificazione Originale

L'immagine in basso mostra uno schema semplificativo della pianificazione originaria.

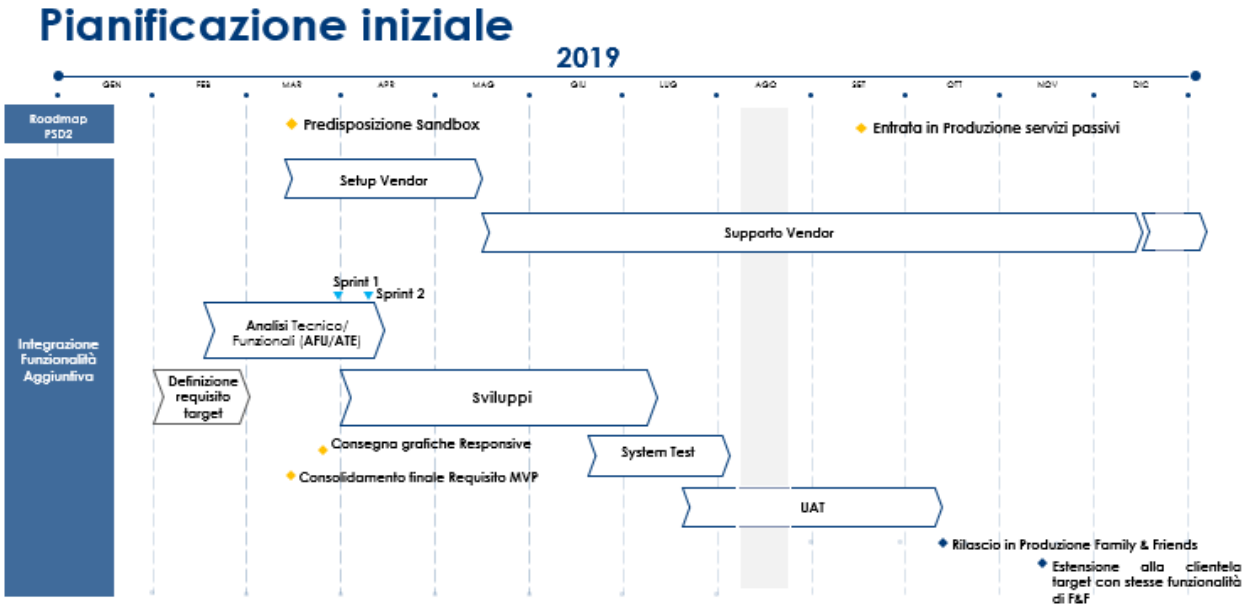


Figure 22 Pianificazione iniziale

I lavori per il progetto sono partiti a Gennaio 2019. Nel mese di Febbraio 2019 si è definito il requisito target. A partire dagli obiettivi strategici identificati dalle divisioni strategiche della banca, è stata definita la visione del prodotto software, le sue principali caratteristiche in termini di funzionalità, gli obiettivi del prodotto in termini di soddisfazione di bisogni e aspettative della clientela target.

Nel mese di febbraio è stata inoltre selezionata la Fintech deputata all'erogazione dei servizi nell'ambito della nuova funzionalità.

Fra marzo e febbraio si sono svolte le analisi Tecnico Funzionali per la redazione dell'AFU, documento che contiene i requisiti tecnico funzionali del prodotto.

A marzo si è quindi definito il perimetro funzionale di prima release (MVP), il master plan di progetto e sono state avviate le fasi operative: le fasi di sviluppo ed integrazione delle componenti IT.

Il progetto oggetto del caso di studio è un progetto dove:

- il contesto è instabile e poco maturo
- i requisiti possono subire variazioni, anche frequenti
- le risorse sono disponibili e competenti
- l'organizzazione è gerarchizzata
- operano diversi contributori, localizzati in differenti sedi
- il progetto software non richiede tempistiche di esecuzione lunghe
- tecnologie e processi sono noti
- è richiesto un time to market breve.

Alcune delle caratteristiche elencate sono più adeguatamente gestite attraverso metodologie tradizionali, come la necessità di documentazione e la composizione del team di progetto, ad altre, come l'incertezza del contesto, rispondono meglio metodologie agili. Per gestire in maniera appropriata le caratteristiche del progetto non è stato utilizzato un approccio univoco, si sono invece adottate logiche dei metodi agili e logiche che si riferiscono ai metodi più tradizionali.

La decisione di un rilascio in ottica MVP deriva da un approccio della metodologia Agile, ed è espressione del Lean software Development. La decisione consente di rispondere a due caratteristiche del progetto: l'incertezza derivante dal contesto PSD2 ancora immaturo e la necessità di garantire time to market brevi, indispensabile per essere competitivi in settori innovativi.

Si è deciso di gestire l'incertezza relativa alla progettualità adottando approcci agili, non solo nella decisione di un rilascio MVP, ma anche nell'impostazione della pianificazione e dello scheduling. Accurate e adeguate informazioni sono disponibili per le fasi temporalmente più vicine per cui la pianificazione è dettagliata, risulta invece accennata per le fasi temporalmente più lontane, su cui non si possiedono in formazioni ancora dettagliate. La pianificazione viene quindi rivista e dettagliata con il progresso delle attività progettuali e la diminuzione dell'incertezza.

Questo tipo di approccio viene definito elaborazione progressiva o Rolling Wave Planning. Il Rolling Wave Planning è un metodo di schedulazione utilizzato nella Metodologia Agile, come lo Scrum, che suddivide la pianificazione in wave, in macro-fasi e si basa sull'assunto che si posseggano informazioni dettagliate sulle attività delle wave temporalmente vicine che consentono una pianificazione di dettaglio e viceversa non si posseggano in formazioni accurate in merito ad attività a lungo termine. (Larman, 2004) La pianificazione di dettaglio delle wave successive è perciò eseguita ad intervalli più o meno regolari.

Le pianificazioni di dettaglio definite sono quindi quelle che guardano ad obiettivi di breve termine. Infatti, se si guarda allo schema della pianificazione iniziale, in basso, le attività per cui è definito il dettaglio sono quelle che riguardano la definizione e l'analisi del requisito. Sviluppi e attività di test sono indicate ma non dettagliate. Sono però definite le milestone progettuali: la data di apertura alla clientela è prevista per metà novembre, a seguito dell'entrata in Produzione dei servizi passivi dell'Open Banking per tutti gli Istituiti finanziari prevista dalla normativa per il 14 Settembre.

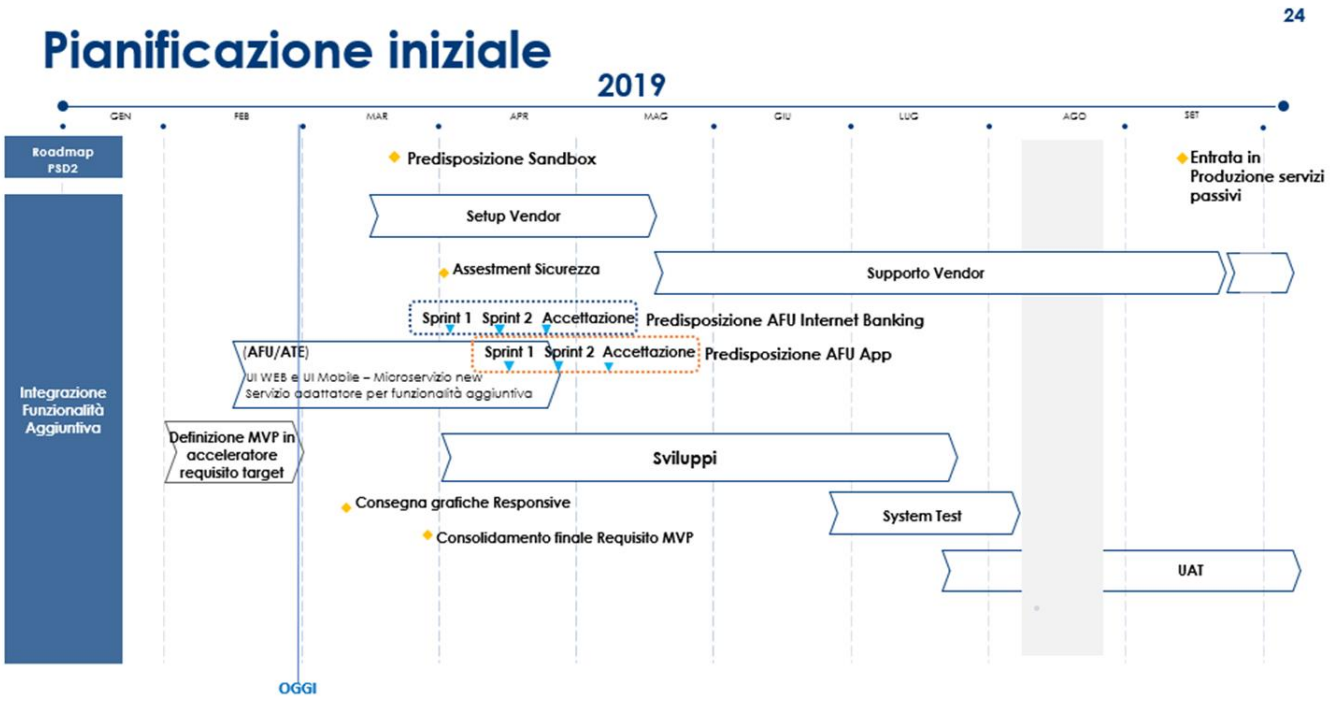


Figure 23 Dettaglio pianificazione iniziale

Per la definizione del requisito e l'analisi tecnica funzionale si sono adottate logiche agili. La definizione del requisito è avvenuta in "Acceleratore" nel mese di Febbraio 2019, in cui è stato prodotto il Business Requirement Book.

Le attività sono state svolte in un'unica sede, da un team piccolo e cross funzionale costituito da utenti, esponenti delle aree strategiche e del business della banca, esponenti della Direzione dei Sistemi Informativi della banca, fra cui il capo progetto e alcuni consulenti a supporto.

Il team ridotto e la localizzazione unica abilita l'utilizzo di metodi agili per la definizione del requisito e le analisi tecnico funzionali. Vengono pianificati tre sprint: il primo della durata di sei settimane, gli ultimi due, della durata di due settimane. L'approccio agile consente di ottimizzare i tempi ma anche e soprattutto l'efficacia dei flussi comunicativi, essenziale nella fase di definizione dei requisiti.

Il prodotto delle attività svolte è costituito da due documenti: l'AFU, il documento di analisi funzionale redatto dagli analisti funzionali e l'ATE, il documento di analisi tecnica redatto dagli sviluppatori.

Le attività svolte in acceleratore riguardano l'analisi tecnica funzionale del requisito ed in particolare:

- UI WEB e UI Mobile[9]: il progetto delle interfacce per i canali Internet Banking e App Mobile
- La definizione funzionale e tecnica dello sviluppo di nuovo microservizio
- La definizione funzionale e tecnica dello sviluppo del servizio adattatore per funzionalità aggiuntiva

[9] User Interface: Interfaccia utente

Il Vendor fornisce la funzionalità aggiuntiva che deve essere integrata nei sistemi della banca, il focus principale degli sviluppi interni è quindi lo sviluppo di funzionalità che siano in grado di integrare i servizi offerti dal Vendor e la progettazione e lo sviluppo delle interfacce.

Il team di sviluppatori ha una conoscenza approfondita dei processi di integrazione. Trattandosi di attività note si è scelto un approccio più tradizionale di gestione del processo di sviluppo a livello di macro-pianificazione, ovvero di definizione del criterio di transizione fra le varie fasi che costituiscono il processo di sviluppo software. Si è adottato un V- Model.

Il modello tradizionale è adatto a contesti gerarchizzati e strutturati, che richiedono un'accurata documentazione di tutte le attività come quello bancario.

La decisione di non adottare un approccio agile anche a livello macro-progettuale è dettata dall'esistenza di diversi vincoli, primo fra tutti quello di avere un team di sviluppo costituito da cinque differenti fornitori localizzati in diverse sedi del Nord Italia.

Il V-Model prevede che le varie fasi seguano un ordine sequenziale, ovvero che una fase non possa esser iniziata se non si è completata la fase che la precede. Nel caso oggetto di studio l'adozione di questo approccio non è rigida. Fra le fasi legati da una relazione del tipo Finish-to-Start, ovvero fra fasi successive, esistono delle sovrapposizioni.

Per ridurre ulteriormente il Time to market, gli sviluppi sono stati affidati a quattro team differenti che hanno sviluppato parallelamente diverse componenti software, si è utilizzato, quindi un approccio di tipo Concurrency.

La divisione delle attività in sub attività con la conseguente attribuzione delle sub attività a team diversi che lavorano in parallelo è stata utilizzata anche per le attività di Testing.

La sovrapposizione dell'attività e l'adozione di un approccio Concurrency è dettata dall'esigenza di garantire time to market brevi, infatti la Concurrency aiuta a diminuire i tempi necessari di consegna del progetto ed introduce un elemento di flessibilità nel modello. Se da un lato consente di ridurre il time to market, dall'altro lato la Concurrency introduce un elemento di complessità nel modello.

In questo caso il progetto è caratterizzato da un grado di incertezza che dipende da fattori esterni e che non diminuisce velocemente nello svolgersi delle attività di sviluppo prodotto, infatti gran parte dell'incertezza deriva dalla maturità dei servizi PSD2 degli istituti bancari o di credito che operano nel settore dei servizi finanziari italiani, questa incertezza può esser risolta solo testando i servizi degli altri istituti. Il test tuttavia, può avvenire solo nelle fasi finali del processo di sviluppo software.

All'interno del Project Charter[10], il team di Progetto ha inserito fra gli assunti l'effettiva disponibilità degli ambienti di Test e di produzione dei servizi PSD2 di Istituti finanziari esterni, spesso indicate come parti terze.

Gli assunti sono elementi che influiscono su tutti gli aspetti del progetto e che vengono ritenuti veri a livello progettuale, anche se non si dispone prove della loro veridicità. Gli assunti implicano un certo livello di rischio e possono rivelarsi spesso infondati. I rischi con le peggiori conseguenze sul

[10] Documento di avvio di progettto, contiene le informazioni relative agli obiettivi progettuali, al contensto di riferimento e agli stakeholder, ai vincoli e agli assunti relativi alla progettualità.

progetto spesso derivano da assunti disattesi. All'interno della Matrice dei Rischi viene riportato il rischio relativo a questo assunto ed il relativo grado di impatto. La gestione del rischio e le azioni per mitigarlo sono tuttavia affidate al fornitore, alla Fintech a cui spetta gestire l'incertezza relativa al contesto PSD2.

L'approccio agile è un approccio flessibile che ha come vantaggio principale quello di gestire l'incertezza. L'incertezza della progettualità è soprattutto legata al contesto PSD2 e potrebbe portare a modifiche e richieste di cambiamenti. Tuttavia, a gestire l'incertezza relativa al PSD2 è il Vendor.

Quindi se il Vendor si occupa di gestire le turbolenze derivanti dal mondo PSD2, la banca dispone di un'architettura IT flessibile, basata sulle API che è in grado di sostenere continue modifiche e facilita la gestione dei cambiamenti.

Prima del rilascio alla clientela, è previsto un rilascio in produzione in modalità Family & Friends a fine ottobre.

Il prodotto viene quindi rilasciato in Produzione per svolgere delle attività di Beta Testing. Il Beta Testing è un test del software eseguito da utenti reali, può essere attivato in diverse modalità fra cui quella Family & Friends, ovvero destinando il rilascio a solo amici e famigliari del team di progetto.

Il Project Charter riporta quindi fra i vincoli della progettualità:

- Vincoli temporali, identificati nelle milestone di rilascio di metà ottobre
- Vincoli economici per cui il budget di progetto è quello definito e approvato in di demand
- Vincoli tecnologici, dettati dalla possibilità di comunicazione fra i sistemi della banca e i sistemi delle parti terze attraverso la Fintech
- Vincoli normativi relativi alla PSD2

3.1.3 Evoluzione della Pianificazione

Nel mese di Luglio 2019 la Banca ha deciso di interrompere i rapporti con la Fintech. La decisione è dovuta a fattori esterni alla progettualità ma risulta impattante sulla pianificazione.

Si è inizialmente svolta una valutazione dell'impatto dell'ingaggio di un fornitore in grado di fornire gli stessi servizi della Fintech precedente. Le procedure per lo scouting e l'ingaggio sarebbero risultate costose e avrebbero comportato un fermo delle attività non sostenibile dalla progettualità, il fermo delle attività infatti, avrebbe coinvolto non solo il team interno ma anche i fornitori esterni.

Per limitare gli impatti dovuti al cambio di Vendor in termini di costi e tempi, si è deciso di sviluppare internamente parte dei servizi precedentemente resi disponibili dal vecchio Vendor e per la parte mancante di avvalersi dei servizi offerti di un nuovo Vendor. Si avviano quindi le procedure per ingaggiare una Fintech analoga, in grado di fornire il delta mancante.

Tutto ciò avviene durante il mese di luglio 2019, periodo in cui sono in fase di finalizzazione i test di integrazione delle componenti IT e sono in corso le attività di System Test. La data di avvio degli UAT è invece, ancora da definire.

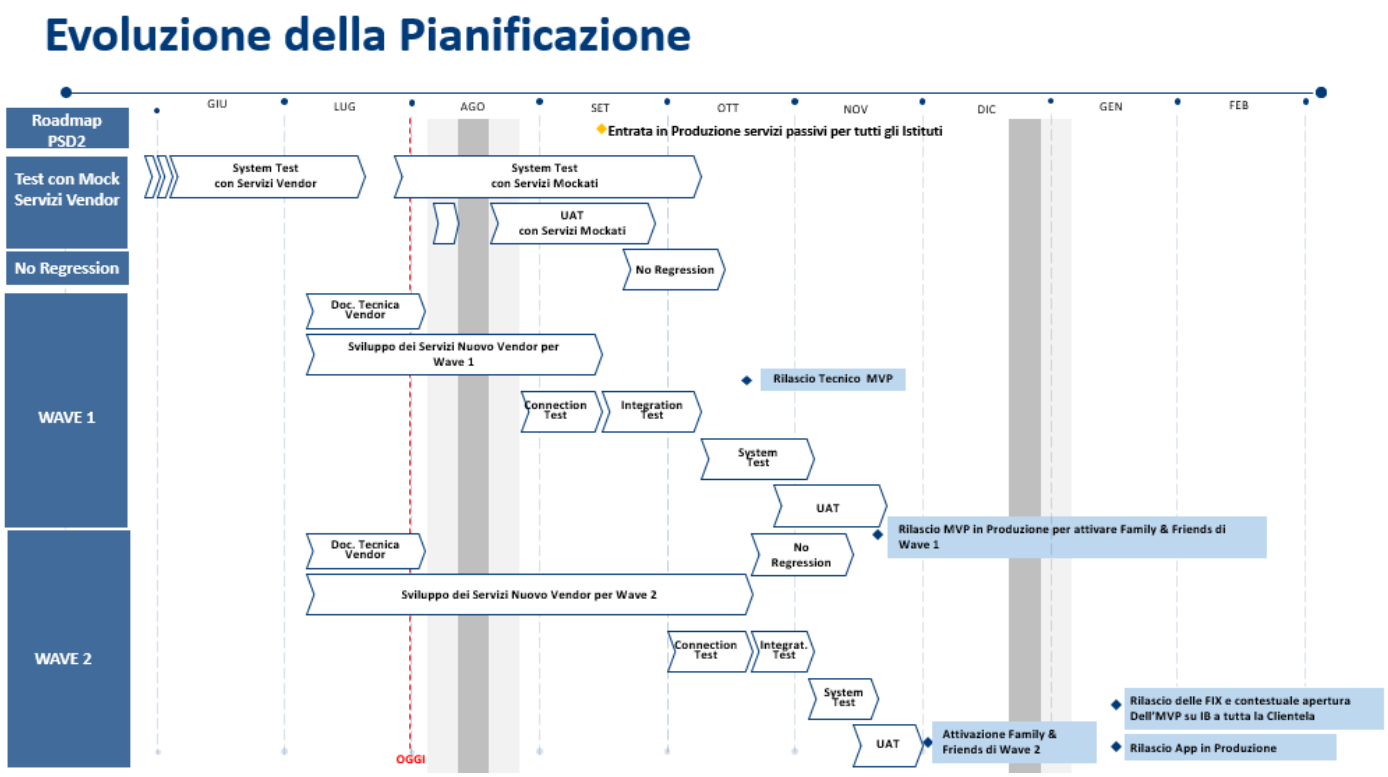


Figure 24 Evoluzione della Pianificazione

Le attività subiscono una ri-pianificazione e il rilascio finale del prodotto sulla clientela prima previsto per metà novembre 2019, viene spostato agli inizi di gennaio 2020.

Come si è detto in precedenza a Luglio 2019 erano in corso i System Test, affidati ad un fornitore. Per evitare il blocco delle attività e per sfruttare al meglio i tempi a disposizione si è deciso di continuare le attività di system test, utilizzando dei servizi mockati. Ovvero, poiché i servizi del primo vendor in grado di fornire la funzionalità aggiuntiva non erano più disponibili, si è deciso di simulare la loro azione attraverso dei Mock e di continuare le attività di test, per verificare il funzionamento delle specificità di front end sia su Internet Banking e che su App. dettagliate in seguito.

La sostituzione del Vendor comporta quindi la necessità di sviluppi aggiuntivi da parte del team di progetto, questo non solo aumenta il carico di lavoro interno ma anche le complessità gestionali e tecniche del progetto.

Le complessità introdotte con la nuova soluzione portano ad una riflessione sull'approccio di gestione da utilizzare e lo spostamento verso logiche più agili. Gli sviluppi interni della funzionalità determinano due elementi di complessità:

1. Due team differenti con differenti approcci, localizzati in due differenti città italiane e provenienti da diverse culture aziendali devono sviluppare congiuntamente alcune funzionalità. La complessità in questo caso è quella relativa all'organizzazione del lavoro e all'implementazione di canali di comunicazione adeguati. Per assicurare l'efficacia dei flussi comunicativi, il fornitore e i team di sviluppo del progetto svolgono meeting giornalieri, della durata di circa 45-60 minuti. Lo scopo dei meeting

è quello di aggiornarsi sul progresso delle attività rispetto al meeting precedente, di segnalare eventuali criticità e di indirizzarne la risoluzione.

2. L'incertezza relativa al mondo PSD2 deve essere gestita congiuntamente dal nuovo fornitore e dal team di sviluppo interno.

Il rischio maggiore identificato per la progettualità è quello relativo all'assunto sulla disponibilità degli ambienti di test e produzione della PSD2 delle parti Terze. Se originariamente questa complessità era gestita esclusivamente del fornitore, con la nuova soluzione il rischio è condiviso fra il nuovo Vendor e il team di progetto. È quindi necessario definire delle opportune misure di mitigazione.

Il rischio è determinato da fattori esterni alla progettualità che non possono essere influenzati o modificati, quindi è ineliminabile. Il rischio si basa su un assunto, ovvero sul ritenere vero qualcosa che potrebbe non esserlo. Una prima misura per mitigare gli impatti del rischio è quella di verificare il prima possibile la veridicità dell'assunto.

Le attività di System Test, svolte prima della ri-pianificazione e con il supporto della prima Fintech ingaggiata, avevano consentito una verifica iniziale in merito alla disponibilità degli ambienti di Test delle parti Terze. I primi test avevano rilevato che non tutte le parti terze testate avevano predisposto adeguati ambienti di test, in termini di disponibilità ma anche di conformità agli standard imposti dalla PSD2.

Le evidenze riscontrate in ambiente di Test hanno contribuito a far nascere l'esigenza di una verifica il più possibile tempestiva della disponibilità degli ambienti di produzione. A tal fine si decide di anticipare il rilascio in produzione di alcune funzionalità del prodotto software verso metà ottobre, un mese dopo la data ultima imposta dalla normativa dell'obbligo di rendere disponibili l'accesso alle informazioni dei dati finanziari dei clienti, previo il loro consenso, alle parti terze autorizzate. Le funzionalità oggetto di rilascio hanno quindi la possibilità di interagire con gli ambienti di Produzione delle parti terze e verificarne la disponibilità e la conformità agli standard PSD2.

Si è deciso di adottare un approccio iterativo ed incrementale, dividendo la pianificazione in due wave e pianificando tre differenti rilasci.

Il primo rilascio in Produzione è pianificato allo scopo di verificare gli ambienti di Produzione delle parti terze, infatti avviene prima che abbiano inizio i system test e gli UAT della prima wave e segue immediatamente dopo la verifica e la validazione dell'integrazione dei servizi della Fintech nei sistemi della banca, ovvero in seguito ai Connection e Integration Test.

Il secondo rilascio in produzione riguarda una parte del perimetro funzionale definito per l'MVP, detto perimetro di Wave 1. Le funzionalità scelte sono quelle per cui i requisiti sono caratterizzate da un minor grado di incertezza rispetto alle funzionalità che andranno a costituire la seconda wave.

Il rilascio in Produzione di Wave 1 è previsto per fine novembre. In seguito, viene attivato un beta testing in forma Family & Friends.

Il rilascio di Wave 2 è previsto invece per metà dicembre e anche a questo è correlato un beta testing in forma Family & Friends.

Solo dopo la validazione del Beta Testing sia di wave 1 che di wave 2 è prevista l'apertura alla clientela.

Le attività di sviluppo, system test e UAT per le due wave sono svolte in parallelo, per garantire un time to market il più possibile breve e cercare per quanto possibile di assorbire il ritardo dovuto al cambio di vendor.

3.1.4 Il piano finale

I primi Test in produzione hanno rivelato che non tutte le parti terze avevano reso disponibili gli ambienti di Produzione PSD2 delle parti terze e in alcuni casi, se disponibili, gli ambienti non rispettavano gli standard tecnici indicati dalla normativa. Spesso inoltre, sono state riscontrate difformità fra ambiente di Produzione e ambiente di Test. Infatti, gli ambienti di test servono a simulare il funzionamento degli ambienti di produzione e per tanto devono esserne una copia fedele, le verifiche hanno rivelato che in realtà non sempre gli ambienti di test riproducevano fedelmente il funzionamento e le specifiche dei rispettivi ambienti di produzione.

Le verifiche hanno quindi mostrato che l'assunto non è stato disatteso, almeno non completamente.

L'impatto delle difformità riscontrate rispetto a quanto atteso è stato comunque rilevante. La funzionalità è stata rilasciata in produzione e aperta alla clientela a fine gennaio 2020, quindi solo qualche settimana più tardi rispetto a quanto previsto dalla nuova pianificazione. Ma il rilascio a fine gennaio è stato possibile soltanto decidendo di limitare ulteriormente il perimetro funzionale previsto per l'MVP.

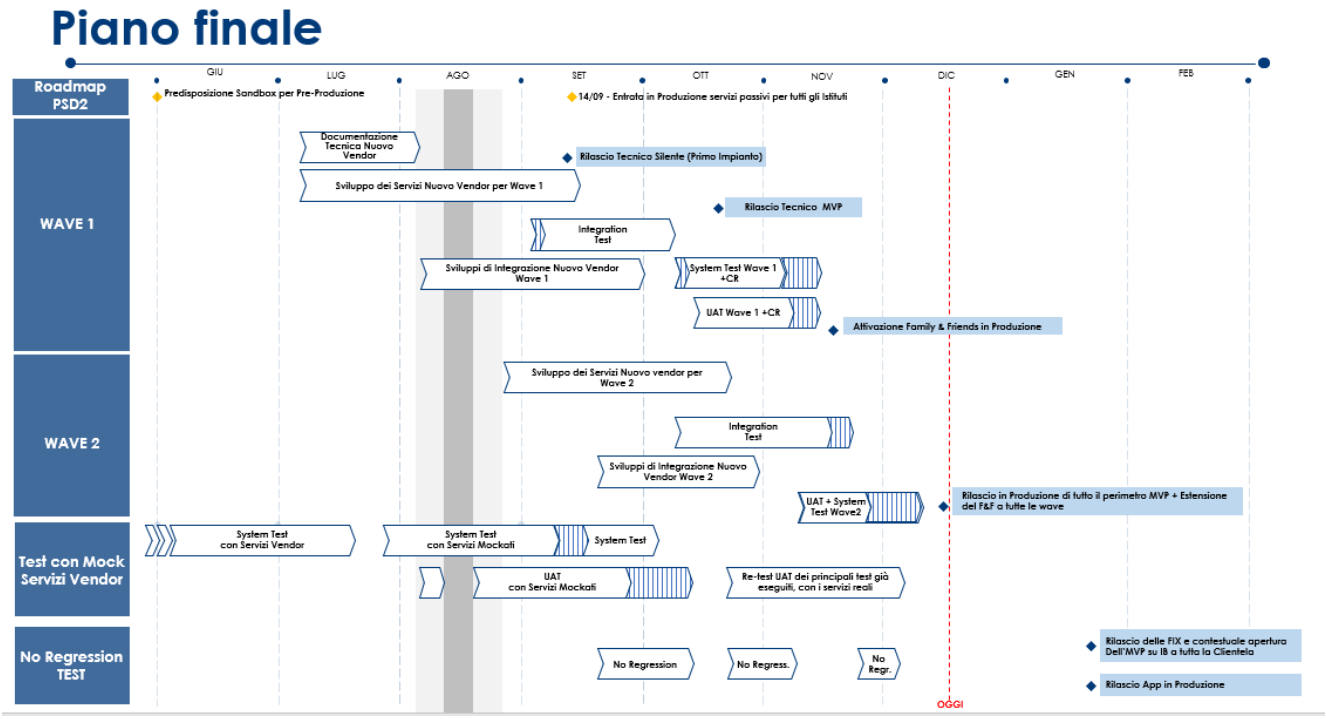


Figure 25 Piano finale

In alto è riportata la pianificazione finale. Con il tratteggio sono evidenziati i ritardi che hanno caratterizzato alcune delle attività. Si può notare come le attività che hanno subito maggior ritardo siano quelle di test.

3.2 LE ATTIVITÀ DI TEST

Per rimanere competitivi nel contesto dell'Open Banking è necessario che le Banche rispondano in maniera tempestiva ai cambiamenti del settore, ma garantire time to market brevi non è sufficiente. Le Banche devono essere in grado di soddisfare i bisogni e le esigenze degli utenti e soddisfare determinati standard di qualità.

Nel caso oggetto di studio, la Banca è parte di uno dei maggiori gruppi bancari italiani e ha quindi una brand reputation da difendere ed è per questo che garantire elevati standard qualitativi del prodotto è di cruciale importanza.

In quest'ottica assumono un'importanza fondamentale le attività di Test le quali servono a verificare e validare la qualità del prodotto software e la sua capacità di soddisfare i requisiti degli utenti. Le attività di Test devono esser pianificate e svolte in modo da rilevare tutti i possibili difetti di un software e rilevarli sin dalle prime fasi progettuali, in modo da poter esser corretti e ritestati sin da subito e ridurre in questo il costo relativo alla correzione degli errori.

Le attività di UAT hanno lo scopo di verificare che il software sviluppato sia conforme ai requisiti. Quale sia la metodologia di sviluppo software più adeguata che aumenti la probabilità di ottenere software conforme ai requisiti è oggetto di discussione e di dibattito

3.2.1.1 Perché il Testing

Il test del software è un processo che consente di identificare la correttezza, la completezza e la qualità del software sviluppato. Lo scopo del test è quello di trovare errori nel software, in modo da correggerli prima di rilasciare il prodotto agli utenti finali. In parole semplici, il test del software è un'attività per verificare che il sistema software sia privo di difetti.
L'attività di Software Testing consiste nel controllare che i risultati attuali corrispondano ai risultati attesi e ad assicurarsi che all'interno del software non siano presenti defect.
Il Testing consiste nell'esecuzione di una componente del software o del sistema per valutare una o più proprietà di interesse; aiuta inoltre ad identificare gli errori, le differenze o le parti mancanti rispetto al requisito definito.
L'attività di test può essere eseguita manualmente o utilizzando strumenti automatizzati.

I Sette Principi Del Test Management
Per assicurarsi che le attività di Test vengano condotte con efficienza bisogna progettare un'adeguata Strategia di Test. I sette principi del software test possono essere di supporto allo scopo.

1. Exhaustive testing is not possible. Testare in maniera esaustiva non è possibile, è invece necessario stabilire un numero ottimale di casi di test, che consenta di rilevare i defect più critici

2. Defect Clustering. La gran parte dei defect rilevati dalle attività di test è contenuta in un numero di moduli limitato, quindi è importante identificare questi moduli per concentrare su di essi le verifiche.
3. Pesticide Paradox. Se si ripete uno stesso test più volte, sarà sempre più difficile che questo consenta la rilevazione di nuovi defect nel software, bisogna quindi variare e migliorare continuamente i metodi di indagine utilizzati.
4. Testing shows a presence of defects. Il software testing deve ridurre la probabilità che difetti non scoperti rimangano all'interno del software.
5. Absence of Error – fallacy. L'attività di software testing non si limita a trovare difetti ma anche a verificare che il prodotto software corrisponda ai requisiti del business.
6. Early Testing. L'attività di test dovrebbe iniziare il prima possibile nel ciclo di vita di un software, in modo che ogni difetto nei requisiti o nella progettazione sia rilevato nei primi stadi. È infatti molto meno costoso aggiustare un defect nei primi stadi dell'attività di testing.
7. Testing is context dependent. L'attività di test è fortemente influenzata dal contesto e quindi in virtù di questo sono richiesti differenti approcci e metodologie. (7 Software Testing Principles: Learn with Examples, n.d.)

3.2.2 I diversi approcci al test

3.2.2.1 Modelli lineari.

Nei modelli sequenziali la Test Strategy, il Test Plan con i relativi Scenari e Casi di Test sono elaborati nelle fasi iniziali di raccolta e analisi dei requisiti. Le attività di Test vengono svolte al termine degli sviluppi, al fondo della catena che definisce la sequenza delle attività.

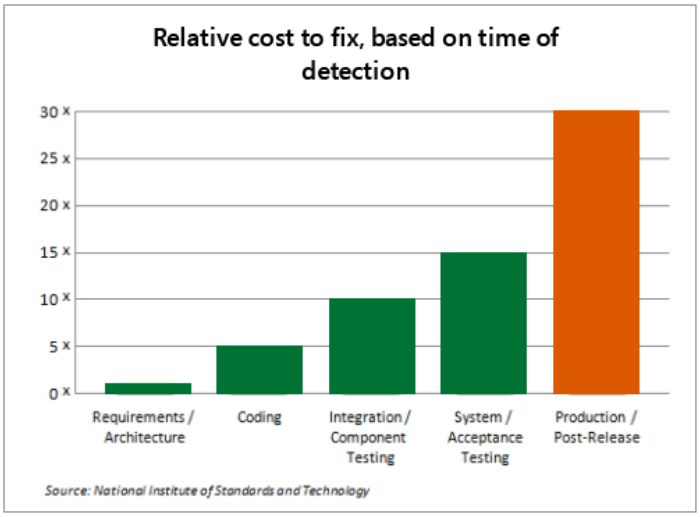


Figure 26 Costo della risoluzione del defect in base alla fase di rilevazione

Se si segue questo approccio il rischio è di rilevare difetti nel software tardi, in questo caso la loro correzione può rivelarsi molto costosa. Più tardi viene rilevato un defect, infatti, maggiore sarà il costo associato alla sua risoluzione, poiché saranno richieste maggiori modifiche per la sua risoluzione.
Vi è inoltre il rischio, di consegnare un prodotto non rispondente alle esigenze degli utenti oppure non conforme ai requisiti se questi non sono stati correttamente specificati o compresi.

Ulteriore problema del modello sequenziale è che le attività di testing possano creare un collo di bottiglia al temine del processo in quanto costituiscono una delle fasi maggiormente effort e timing intensive dello sviluppo software.

Figure 27 Differenti tipologie di testing

3.2.2.1.1 UNIT TESTING

Le attività di Unit Testing hanno lo scopo di verificare il corretto funzionamento dei singoli moduli che costituiscono le componenti di un prodotto software. Dato un insieme di input, lo scopo è assicurarsi che il modulo restituisca l'output atteso e corretto. Ad eseguire le attività di Unit Testing sono gli stessi sviluppatori.
Gli Unit Testing devono essere eseguiti ogni volta che viene implementato un nuovo codice, il codice costituente un modulo viene modificato o ci sono cambiamenti nel codice di altri moduli.
Gli Unit Testing abilitano il parallelismo fra i test dei singoli moduli che costituiscono i componenti del software, i quali possono essere testati in parallelo dagli sviluppatori e indipendentemente dal fatto che sia concluso lo sviluppo di tutti i moduli. Tuttavia, si possono riscontrare delle difficoltà nella loro esecuzione nel caso di test di moduli il cui funzionamento non è indipendente bensì correlato.
Gli Unit Testing sono economicamente efficienti e consentono di rilevare e rimuovere i difetti ad un costo inferiore rispetto a quanto avviene nelle altre fasi di test.
(Unit Testing, n.d.)

3.2.2.1.2 INTEGRATION TESTING

Lo sviluppo di un prodotto software, soprattutto nei casi in cui è ampio e complesso, avviene dividendo il prodotto in singole componenti o moduli, il cui sviluppo è spesso implementato da diversi team o sviluppatori che lavorano in parallelo. Il funzionamento dei singoli moduli è verificato in fase di Unit Testing.
Una volta che i moduli sono sviluppati e verificati è necessario procedere alla loro integrazione, ovvero bisogna porre i moduli in comunicazione fra loro per costruire il sistema software. La comunicazione dei moduli avviene attraverso delle interfacce. Per verificare che l'integrazione fra i moduli sia avvenuta, cioè che i moduli siano in grado di comunicare fra loro a prescindere dalla bontà e dalla correttezza della comunicazione in sé, è necessario eseguire degli Integration Testing.
(Integration Testing, n.d.)

3.2.2.1.3 SYSTEM TESTING

Una volta verificata l'integrazione fra le componenti software, il software può essere eseguito come sistema e si possono eseguire le attività di system test.

Il System Test consiste nell'esecuzione di diverse attività che consentono di verificare il sistema completo nel suo funzionamento e nelle modalità con cui il prodotto software si interfaccia alle componenti software e hardware ad esso correlate.

Ogni aspetto del sistema è testato: considerato ogni possibile input si verifica che il sistema in ogni sua parte risponda con l'output atteso. Durante le attività di System Test viene verificato il funzionamento di ogni applicazione integrata che si interfaccia con il prodotto software, come queste interagiscono fra loro e con il sistema.

Diversi sono i test che fanno parte della famiglia dei System Test, ciascuno volto a verificare determinati aspetti del sistema, quali le performance del software quando il sistema è sottoposto ad un carico reale (Load testing), l'interazione fra i componenti hardware e software oppure l'affidabilità dell'applicazione in caso di arresti anomali (Recovery testing). Altre tipologie di software Testing sono:

- Usability Testing: riguardano la facilità d'uso del prodotto e la sua capacità di soddisfare le esigenze degli utenti
- Regression Testing: aggiunte o cambiamenti all'interno del prodotto software possono causare dei malfunzionamenti, il Regression testing ha lo scopo di verificare che tali malfunzionamenti siano assenti
- Functional Testing: sono volti a prevedere possibili miglioramenti dell'applicazione in termini di funzionalità. Verificano quindi che il set di funzionalità offerto dal prodotto software sia completo e in grado di soddisfare pienamente gli scopi del prodotto

Le differenti attività di System Test possono essere svolte nella loro totalità oppure si può decidere in fase di Test strategy di eseguirne un sottoinsieme. Le valutazioni sul numero di tipologie di system test da eseguire e sul tipo di test necessari riguardano diversi aspetti. Nella decisione vanno considerati vincoli in termini di budget e tempi (più test comporteranno maggiori costi e tempistiche più lunghe), di risorse (in caso si disponga di strumenti che consentano di automatizzare le attività di test le tempistiche si riducono) ma anche il contesto aziendale e le competenze dei tester. (System Testing, n.d.)

3.2.2.1.4 USER ACCEPTANCE TESTING (UAT)

Lo User Acceptance Testing è il test eseguito dagli utenti finali per verificare il prodotto software prima del suo rilascio in Produzione.
Il test di UAT viene solitamente eseguito in un ambiente che ha una configurazione simile all'ambiente di Produzione.
Lo scopo principale dell'UAT è quello di verificare che il prodotto sia in grado di soddisfare i requisiti per cui è stato progettato, esporre problemi di funzionalità di business che i test unitari e i test di sistema hanno tralasciato, in quanto il focus dei test unitari e di sistema sulle logiche funzionali è minimo.
Il processo attraverso cui vengono svolti i test di UAT consiste in diverse attività: l'analisi dei requisiti, la creazione del piano di test, l'identificazione di scenari UAT e dei di casi di test, la preparazione dei dati necessari all'esecuzione, l'esecuzione dei test e la registrazione dei risultati.

Table 2 Fasi delle attività di UAT

Fase	Obiettivo	Deliverable Prodotti
Analisi dei requisiti	L'analisi dei requisiti è utile per la determinazione di quali siano le funzionalità che sono in grado di soddisfare i requisiti aziendali e che quindi devono essere testate.	-
Creazione del piano di UAT	ll piano di test UAT delinea la strategia utilizzata per verificare e garantire che il software soddisfi i requisiti aziendali.	Test Plan Test Plan
Identificazione Test Case e Test Scenario	Identificazione dei Casi di test e dei Test Scenario	Test Scenario Casi di test
Data Preparation e Bed Preparation	Preparazione dell'ambiente di test e dei dati su cui eseguire i casi di test	-
Esecuzione dei casi di Test	I casi di test vanno eseguiti e va tenuta traccia dei risultati. Devono poi essere identificati e segnalati gli eventuali defect	Test Result Defect Log
Conferma che il prodotto rispetti gli obiettivi di business	Approvazione del prodotto in vista del rilascio in produzione	Documento approvazione formale

La tabella riportata in alto mostra le attività da svolgere per eseguire i test di UAT. Il primo passo è quindi l'analisi dei requisiti che si svolge analizzando i documenti di progetto quali: Project Charter, Business Use Cases, Process Flow Diagrams, Business Requirements Document (BRD), System Requirements Specification (SRS). Vengono quindi identificati i requisiti e le funzionalità che li soddisfano.
Identificate le funzionalità da testare, si passa alla definizione della Test Strategy e del Test Plan. Si tratta di due documenti il cui obiettivo è complementare e in alcuni tratti coincide, perciò sono spesso confusi oppure accade di frequente che uno sia integrato nell'altro.
Il test plan documenta le funzionalità oggetto di test e la loro corrispondenza ai requisiti analizzati. Contiene inoltre la pianificazione delle attività e l'allocazione delle risorse necessarie per l'esecuzione degli UAT nei tempi determinati.
In un certo senso il Test Plan può essere considerato come l'implementazione operativa della Test Strategy. Infatti, la Test Strategy è costituita dall'insieme delle linee guida adottate da gruppo di lavoro per l'esecuzione delle attività di test in termini di obiettivi e strumenti.
A partire dal Test Plan si definiscono gli Scenari di test e i Casi di test. Gli Scenari di test identificano quale funzionalità testare e sono costituiti da un insieme di casi di test che invece identificano i passi da svolgere per testare una determinata funzionalità. Ciascun passo corrisponde ad un caso di test. Diversi sono gli aspetti che possono essere costituire oggetto di test, quali funzionalità, performance, usabilità. In base alle linee guida definite dalla Test Strategy, vanno progettati casi di test per ogni funzionalità e per ogni aspetto di questa che si vogli testare.
Una volta definiti i Casi di test è necessario predisporre un set di dati, il più possibile rappresentanti la realtà con cui verranno svolti i test. Anche l'ambiente di Test predisposto deve essere il più possibile corrispondente alla realtà. Terminata la Data Preparation, si è pronti per eseguire il test vero e proprio del sistema.

Gli utenti eseguono i Casi di Test nell'ambiente predisposto e tracciano i risultati. Lo scopo dell'esecuzione delle attività di Test è la rilevazione dei difetti che devono essere appropriatamente segnalati. Terminata l'esecuzione dei test se non ci sono difetti critici ancora irrisolti e il prodotto rispetta le aspettative degli utenti, questi procedono all'approvazione formale del pronto che a questo punto si ritiene pronto per il rilascio in Produzione. (User Acceptance Testing, n.d.)
Gli UAT possono essere di due tipologie:

- Alpha Testing: I test alfa vengono eseguiti in ambiente di laboratorio, generalmente da membri da risorse interne all'azienda
- Beta Testing. Il test viene eseguito da utenti reali in ambiente di produzione. Il rilascio del prodotto è rivolto ad un numero limitato di utenti finali, che possono essere risorse aziendali o utenti esterni, che forniscono i feedback rappresentativi della risposta del mercato prima dell'effettivo rilascio alla clientela.

3.2.2.1.5 L'ESECUZIONE DEI CASI DI TEST

Prima che le attività di test possano essere svolte, deve essere progettato un Caso di Test, poi scritto e solo dopo eseguito. I casi di test scritti vengono solitamente conservati in una repository condivisa con il team di test e di sviluppo.
L'esecuzione delle attività di test avviene in uno specifico ambiente, anche detto Bed Test. Il Bed Test è l'ambiente di esecuzione del test configurato per il test. Comprende hardware, software, sistema operativo, configurazione di rete specifici, il prodotto in prova, altri software di sistema e software applicativo (TutorialsPoint, n.d.).
L'esecuzione implica la registrazione dei risultati e la segnalazione di eventuali difetti rilevati in caso di esecuzione. Quando il defect è risolto, il test è rieseguito per verificare l'effettiva risoluzione. Verificato ciò, il test può considerarsi superato. L'esecuzione dei casi di test può avvenire manualmente o attraverso strumenti automatizzati.
Durante l'esecuzione dei casi di test vengono realizzati diversi report per monitorare lo stato delle attività.

3.2.2.2 Modelli iterativi.

Il Testing nel modello iterativo viene eseguito ad ogni iterazione. All'interno della singola iterazione, è sviluppato e immediatamente testato un insieme di funzionalità. Le funzionalità da testare sono relativamente poche e le attività di test vengono eseguite velocemente. Il metodo iterativo ha il vantaggio di consentire di effettuare la correzione, le modifiche o le aggiunte dedotte dai risultati del test, immediatamente nella iterazione successiva.
Grazie all'iterazione dei processi i tester possono migliorare il processo di apprendimento e la velocità di esecuzione delle attività.
Nel modello iterativo importanza fondamentale assumono i test di integrazione e non regressione. All'aumentare del numero di incrementi e iterazioni, aumenta il carico di queste attività di test. In questo caso è utile disporre di strumenti di automazione per lo svolgimento dei test di integrazione e non regressione.

3.2.2.3 Modelli agili.

Nei Modelli agili le attività di test sono eseguite in maniera continua rispetto alle attività di sviluppo e non in modo sequenziale, come accade per il Waterfall.

Nei modelli lineari business analyst, tester e developers sono figure distinte, nel caso dell'agile testing invece esistono delle sovrapposizioni fra i ruoli. L'intero team è il responsabile per le attività di test. Si possono distinguere diverse fasi del testing.
Prima che si compia l'iterazione iniziale si identificano le risorse che avranno la responsabilità principale delle attività di test e gli strumenti, si procede poi all'elaborazione del test plan che viene aggiornato ad ogni iterazione. Ad ogni iterazione vengono condotti dei Confirmatory Test (test che verificano che il sistema è in grado di rispondere alle esigenze degli utenti, eseguito dal team) e l'investigative testing (integration testing, load testing, security testing) solitamente automatizzati.
Il modello agile è un modello iterativo per cui è necessario eseguire continuamente test di integrazione.
Al termine del processo iterativo prima della release, utenti formati allo scopo con il supporto di tester e sviluppatori eseguono un test completo del sistema.

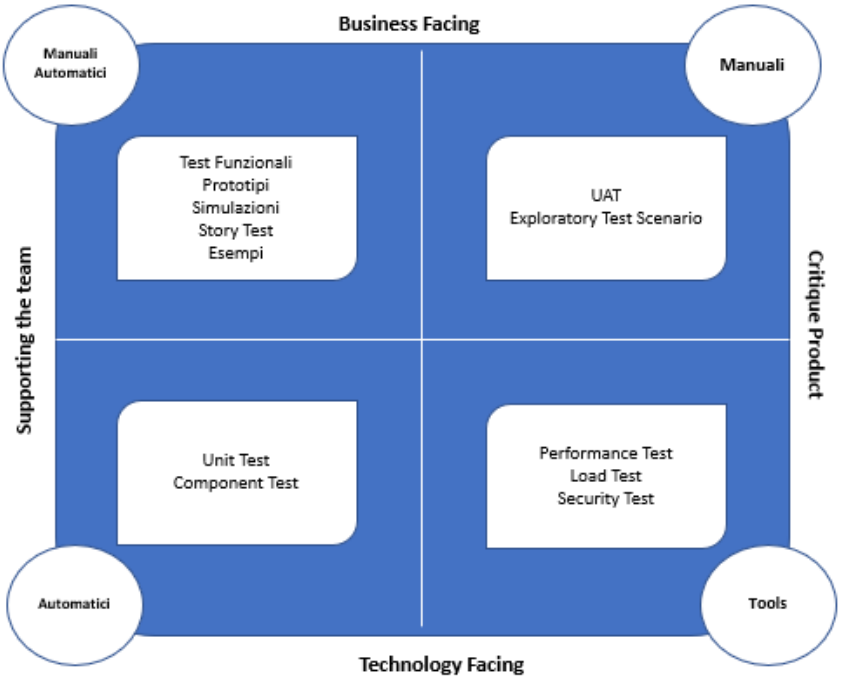


Figure 28 Agile Testing

Il quadrante dell'agile testing aiuta a identificare il metodo di esecuzione delle varie attività di test.

- Quadrante I: lo scopo dei test di questo quadrante è attestare la qualità interna e tecnologica del sistema, sono eseguiti in maniera automatica
- Quadrante II: lo scopo dei test di questo quadrante è attestare la corrispondenza ai requisiti di business, sono svolti sia manualmente che in maniera automatica
- Quadrante III: lo scopo dei test di questo quadrante è attestare la corrispondenza ai requisiti di business, ma anche il corretto funzionamento del sistema. Sono eseguiti manualmente a ogni iterazione
- Quadrante IV: lo scopo dei test di questo quadrante è verificare requisiti non funzionali come sicurezza, performance. Sono svolti in modo automatico.

(Agile Tetsing, n.d.)

La mancanza di documentazione nel testing agili porta ad un maggior rischio di commettere errori. Nei modelli agili esiste una sovrapposizione di ruoli che richiede membri del team di progetto altamente competenti. I modelli agili richiedono inoltre una ridotta finestra temporale disponibile

per le attività di test. Il più grande ostacolo che riscontrano le attività di agile testing è l'introduzione di modifiche e nuove funzionalità, che portano ad una ulteriore riduzione del timing disponibile per i test.

Fra gli approcci agili, troviamo il Test Driven Development, utilizzato in metodi di sviluppo iterativi. Prima della scrittura effettiva del codice sorgente, gli sviluppatori scrivono i casi di test ed eseguono i test per testarne la logica ed il funzionamento. Effettuata la verifica, si passa allo sviluppo del software vero e proprio, ovvero alla scrittura del codice. Quindi, qui, i test guidano lo sviluppo del software viene chiamato in modo appropriato sviluppo guidato dai test: "Test Driven Development".

3.2.3 Il Caso di studio

L'approccio utilizzato per la gestione delle attività di test è un approccio tradizionale che segue le logiche del V-Model.

Il V-Model è un modello plan driven e sequenziale che prevede la definizione dei Casi di Test da eseguire in fase di sviluppo. In particolare, fa corrispondere ad ogni fase di sviluppo software una specifica fase di test, ovvero quando si progetta il software a livello del singolo modulo si scrivono parallelamente i casi di test da svolgere durante la fase di Unit Testing e quelli del System Test durante la fase di progettazione del sistema software.
Le attività di Test vengono quindi eseguite in sequenza al termine degli sviluppi.
Nel mese di Febbraio 2019 si è definito il requisito target, fra marzo e febbraio si sono svolte le analisi tecnico-funzionali per la redazione dell'AFU.
Una volta definiti i requisiti funzionali, si è passati all'identificazione dei Test Scenario e alla scrittura dei Casi di test per gli UAT.

3.2.4 Le attività di UAT

L'UAT è l'ultima fase del processo di test del software e il suo obiettivo principale è garantire che la funzionalità aggiuntiva soddisfi tutti gli user scenario identificati in fase di requisito. Questa fase consente agli utenti finali di completare una revisione finale del sistema prima della release. Solo dopo che l'UAT è stato completato e gli utenti e gli stakeholder sono soddisfatti della soluzione, il sistema può essere approvato.
Lo scopo delle attività di UAT include il test funzionale e non funzionale degli scenari identificati. Ovvero che i requisiti siano soddisfatti e che siano soddisfatti altri criteri di qualità dell'applicazione (ad es. Usabilità del sistema, dati di riferimento e prestazioni del sistema, ecc.)
I Casi di test quindi sono stati scritti in fase di analisi dei requisiti, seguendo i principi del V model.

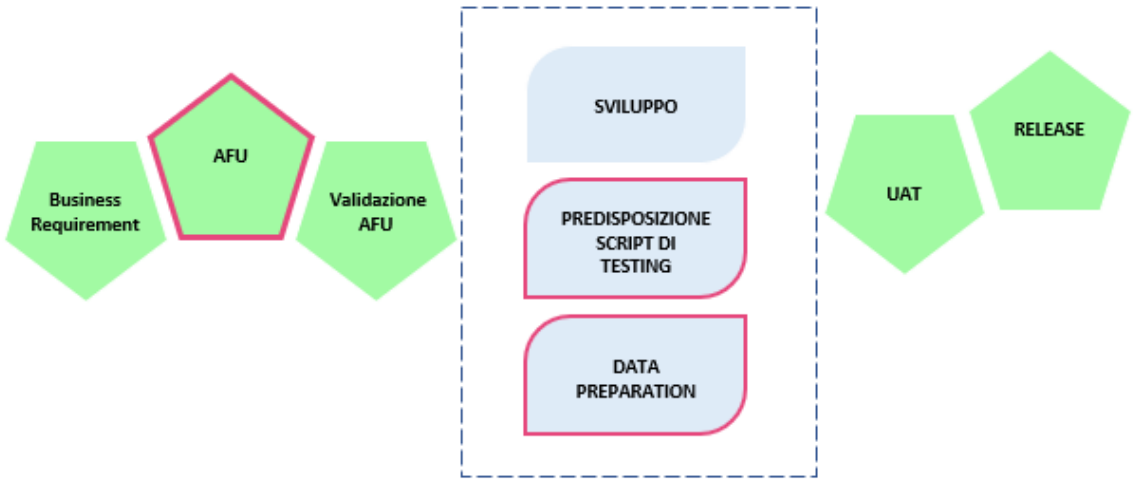


Figure 29 Attività di UAT nel V Model

L'UAT è pianificato e progettato dal team di progetto, ed eseguito e verificato dalla Test Factory della banca per ciò che riguarda il canale Internet Banking e da membri delle aree del business della banca per il canale App Mobile. Il team di Test è quindi costituito da membri esperti con una profonda conoscenza dei sistemi informativi bancari, in grado di assicurarsi che il software sia in grado di gestire le attività richieste in scenari reali. Le attività di Defect management sono state affidate ad un fornitore esterno, già incaricato dello svolgimento delle attività di system test.
Lo strumento utilizzato per la gestione delle attività di test è l'Application Lifecycle Management di HP (HPQC- HP Quality Center). Si tratta di uno strumento Web base che aiuta le organizzazioni a gestire il ciclo di vita di un prodotto software dalla pianificazione del progetto e raccolta dei requisiti, fino alle attività di testing e defect management.

3.2.4.1 UAT Preparation

Per strutturare e classificare l'ambito di accettazione funzionale, sono stati definiti i Test scenario che corrispondono a diversi macro-requisiti dell'applicazione. I macro-requisiti individuati sono cinque, ad ogni macro-requisito corrisponde un test set. Ciascun test set è eseguito quattro volte: per il canale Internet Banking sui due browser Internet Explorer e Chrome e per il canale App sui due sistemi Android e iOS.
I casi di test UAT sono essere definiti anche fra diversi requisiti, garantendo una copertura end-to-end della funzionalità sviluppata.

Gli obiettivi di tale struttura sono:

1. Fornire ai tester UAT una panoramica semplice e comprensibile del progetto in termini di funzioni e funzionalità, disponibilità e stabilità del software
2. Garantire il tracciamento e la documentazione delle attività al momento della revisione del processo e della decisione di accettazione.

L'analisi e la progettazione dei casi di test sono state eseguite principalmente in Excel, mentre l'esecuzione è eseguita in HPQC. Sono stati definiti i seguenti attributi per ogni test e di conseguenza importati da Excel a HPQC:

Table 3 Formalizzazione Casi di Test in HPQC

Attributo	Descrizione

Titolo del caso di test	Titolo del caso di test
Pre-condizioni per il caso di test	Fornisce indicazioni su cosa deve essere fatto prima dello svolgimento del caso di test, contiene informazioni sui casi di test e su possibili interdipendenze fra i casi di test
Descrizione caso di test: procedura	Descrizione del test case, cosa deve fare il tester per eseguire il test case (azione dell'utente)
Descrizione caso di test: risultati attesi	Descrizione dei risultati attesi del test
Team	UAT
Priorità	La priorità viene utilizzata per strutturare il piano di esecuzione
Designer	Nome del progettista del caso di test
Tipo di test	Funzionale- Non Funzionale
Canale	IE- Chrome; Android-iOS

Il risultato della verifica di ciascun caso di test deve essere documentato in modo tracciabile e appropriato. I seguenti stati sono possibili per un caso di Test e devono essere impostati in base alla descrizione:

- Passed: il Test Case viene eseguito e superato con successo
- Failed: non è stato possibile eseguire correttamente il Caso di prova ed è stato aperto un defect
- Blocked: il caso di test non può essere eseguito a causa della dipendenza da un altro caso di test che è Failed o a causa di problemi dell'applicazione (defect). Tutti i casi di test che non possono essere eseguiti a causa della mancanza di correzione da parte delle applicazioni sono impostati su Blocked dal tester
- Not completed: il Test Case è in fase di esecuzione
- N/A: Stato per i passaggi che non possono essere eseguiti nella fase corrente

3.2.4.2 Defect Management

Il defect è una condizione all'interno di un prodotto software per cui questo non rispetta i requisiti del software (così come stabiliti nelle specifiche di requisito).
Il processo di defect management definisce le attività necessarie per identificare e risolvere potenziali defect. I defect sono innescati non necessariamente dal software ma anche da un dettaglio nel requisito, dalla user story, un documento tecnico, casi di test e casi d'uso.
Per gestire la reportistica dei defect nel ciclo di vita del defect stesso si utilizza HPQC. Si definisce ciclo di vita del defect, il flusso di lavoro che un defect subisce durante la sua vita, dalla sua apertura fino alla sua chiusura.
Tutti i difetti UAT devono essere documentati e tracciati in HPQC, secondo le linee guida fornite. I dati inseriti in HPQC sono utili per la gestione del difetto segnalato durante il ciclo di vita del difetto e la valutazione dello stato del progetto, in particolare in termini di qualità del prodotto e progressi dei test.
A seconda del punto del ciclo di vita che occupano lo stato dei defect si distingue in:

- **Aperto.** Il difetto è stato registrato nel Quality Center ma non è stato ancora assegnato. Se il tester ha identificato il gruppo o la risorsa che è in grado di correggerlo, può assegnare direttamente il difetto, impostando lo stato su "Assegnato"
- **Assegnato**. Il difetto è stato assegnato a chi dovrà occuparsi della sua correzione. La risorsa o il team di sviluppo provvederà alla correzione del defect e lo porrà in stato corretto

- **In lavorazione**. Il team di sviluppo ha verificato che la correzione del defect è di sua pertinenza e lo pone in lavorazione
- **Corretto.** Defect risolti. Il tester che li ha rilevati deve rieseguire il caso di test che ha permesso di rilevarli per verificare la correzione. Quando è verificato, il defect è posto come chiuso
- **Respinto al Defect Manager.** Defect assegnati al team di sviluppo, ma la cui risoluzione è eseguibile da altre risorse perché ad esempio, impatta funzionalità diverse da quella gestita dal team a cui è stato assegnato
- **Respinto al Tester**. La risoluzione del defect richiede maggiori informazioni per la risoluzione oppure sono stati analizzati ed è stato confermato che non si tratta di defect.

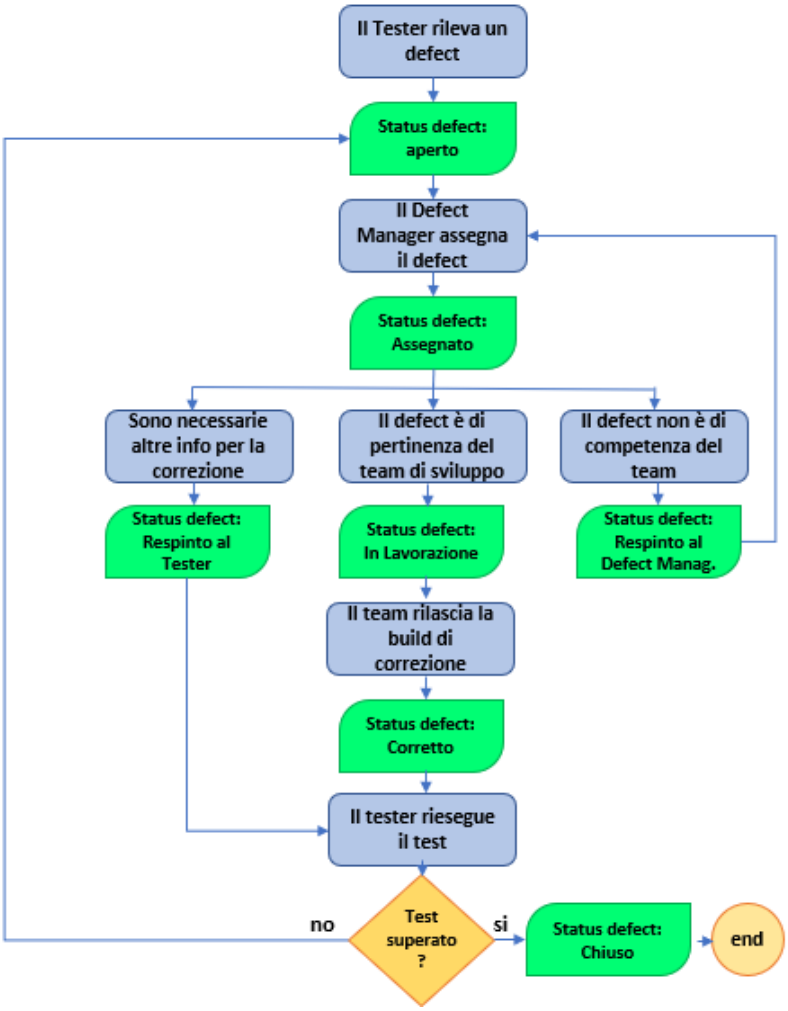


Figure 30 Ciclo di vita del defect

Gli attori del processo che caratterizza il ciclo di vita del defect sono:

- **Il Tester.** Il Tester esegue i passaggi descritti nel Test case e verifica che i risultati ottenuti siano effettivamente quelli indicati nel Test Case. A seconda dei risultati, classifica il test come *passed* o *failed* e riporta lo stato in HPQC. Nel caso di anomalia, il Tester: verifica se esiste un defect analogo in HPQC e immette i difetti in HPQC secondo le linee guida fornite, fornendo una descrizione e l'indicazione su come riprodurlo e l'indicazione sulla priorità. Non appena viene corretto, riesegue il test per verificare la correzione. Se corretto, lo imposta su *chiuso.*
- **Il Defect Manager.** Il Defect Manager ha il compito di assicurarsi che il processo di gestione dei defect funzioni come definito e che i defect siano documentati come richiesto. Gestisce

le riunioni relative ai defect, in particolare due riunioni settimanali e la riunione dell'impatto dei defect non chiusi per il Go-Live, organizza sessioni ad hoc per favorire la risoluzione di difetti molto elevati, tiene traccia e monitora gli indicatori KPI.
- **Il team di sviluppo.** Il team di sviluppo è responsabile della correzione dei difetti assegnati e della descrizione di ciò che è stato corretto. Deve assicurarsi che nella descrizione sia sempre presente un commento una volta che un difetto è stato risolto o rifiutato.

Sono possibili ulteriori classificazioni dello stato dei defect. I defect possono essere classificati in vario modo a seconda del grado di severity e priority.
La priorità è una valutazione del defect in relazione al processo di test effettuata dal Defect manager e viene utilizzata per controllare e gestire il processo di test (avanzamento, pianificazione). Prima del rilascio in produzione, la priorità è utile per stabilire le priorità nella correzione dei defect. La priorità può cambiare nel tempo. Al grado di priorità viene assegnato un indice numerico, utile per il calcolo di alcuni KPI.

- **Very High (0).** Le attività di test e il rilascio non possono procedere finché il test non è risolto**.** È necessario risolverli immediatamente perché stanno bloccando l'esecuzione dei test.
- **High (1).** I defect ostacolano l'esecuzione di casi di test critici e impattano le attività di test di un'intera funzionalità.
- **Medium (2).** Deve essere risolto per consentire la completa esecuzione dei test, ma non incide sull'andamento dei test a livello di pianificazione generale
- **Low (3).** Piccolo o nessun impatto sull'avanzamento del test. Non ha bisogno di essere risolto per il rilascio successivo

La Severity indica la criticità del difetto rispetto al sistema o al processo aziendale. Fornisce inoltre un'indicazione se è possibile o meno un rilascio in produzione con il difetto lasciato non risolto. Il tempo di rilevamento non influisce sulla gravità (solo priorità). Al grado di severità viene assegnato un indice numerico, utile per il calcolo di alcuni KPI.

- **Urgente (0)**. Il difetto impatta su una funzionalità critica, oppure blocca un processo end-to-end o provoca importanti riduzioni delle prestazioni. In questo caso nessuna soluzione alternativa è disponibile e il test case è in stato failed
- **Alta (1)**. Il difetto influisce sulla funzionalità più piccola, non sui sistemi completi, una soluzione alternativa non è disponibile o non è sostenibile nella produzione e il test case è in stato failed
- **Media (2).** Il difetto comporta che il sistemea non funzioni come previsto, ma siste una soluzione alternativa al defect che consente il proseguimento del test. Le soluzioni alternative messe in atto possono essere supportate in un arco di tempo a lungo termine (ovvero mesi). Il test case è in stato passed
- **Bassa (3).** Esistono soluzioni alternative in grado di funzionare fino a quando il difetto non viene risolto o fino a quando una versione futura non corregge il difetto. Le soluzioni alternative (se presenti) hanno un impatto minimo sulle risorse esistenti. Senza queste modifiche, il sistema verrà comunque eseguito correttamente in produzione. Il test case è in stato passed.

In base al grado di severity e a quello di priorità i defect si distinguono in:
- **Bloccanti** per il passaggio in Produzione. Se non risolti sono non può avvenire il rilascio in Produzione.

- **Non Bloccanti** per il passaggio in Produzione. Se non risolti sono può avvenire comunque il rilascio in Produzione.

3.2.5 Analisi delle attività di Test

Nel mese di luglio 2019, periodo in cui sono in fase di finalizzazione i test di integrazione delle componenti IT e sono in corso le attività di System Test. La data di avvio degli UAT è invece, ancora da definire.

3.2.5.1 Monitoraggio

Durante l'esecuzione, sono stati pianificati incontri regolari che hanno coinvolto i team responsabili dell'UAT per discutere i progressi, identificare i rischi e le problematiche e definire le azioni necessarie. La frequenza degli incontri ha subito variazioni in base al numero alla criticità dei defect aperti e all'avanzamento dell'esecuzione, fino al massimo di 2-3 incontri settimanali.

Ad ogni SAL[11] (Stato Avanzamento Lavori) di progetto è stata comunque presentato e discusso lo stato di avanzamento delle attività di test e le criticità legate al test. Per garantire una corretta condivisione delle informazioni ai SAL è richiesta la presenza di tester, sviluppatori e utenti.

È stato previsto un invio giornaliero all'intero team di progetto, dei report sulle attività di UAT ottenuti mediante un'estrazione da HPQC. I report contengono informazioni relative allo stato dei casi di test e dei relativi defect. Il team di PMO ha condotto giornalmente un'analisi della reportistica di dettaglio, ricavando i KPI che consentivano una valutazione di più ampio spettro delle attività di test nel loro complesso. Il risultato delle analisi relativamente all'andamento di KPI predefiniti è stato condiviso con il team di progetto ad intervalli regolari. Le analisi, in particolare, si focalizzano sulle criticità emerse e suggeriscono possibili misure cautelative.

3.2.5.2 Metriche di software testing

Oggi la qualità è ciò che guida la popolarità ed il successo di un prodotto software, il che ha reso necessario prendere le misure necessarie per assicurarsi il raggiungimento di determinati standard di qualità. Per assicurarsi ciò, i tester utilizzano modi standardizzati per misurare i loro obiettivi e la loro efficienza, il che è reso possibile con l'uso di varie metriche di software testing e di KPI. Le metriche ed i KPI svolgono un ruolo cruciale nel determinare l'efficacia e l'efficienza dei team di test e li aiutano a valutare la qualità, l'efficienza, il progresso del test di software.

Le metriche di test del software sono il modo migliore per misurare e monitorare le varie attività di test eseguite dal team di tester. Di seguito sono riportati alcuni KPI solitamente utilizzati nel processo di controllo della qualità di un software.

[11] SAL: Stato Avanzamento Lavori, incontri settimanali del team progettuale per aggiornamenti in merito all'attività di progetto

Table 4 Lista metriche considerate per le attività di Test

KPI	Descrizione	Formula
Test superati	La percentuale di test superati viene valutata monitorando l'esecuzione di ogni ultima configurazione all'interno di un test. Ciò aiuta il team a comprendere l'efficacia delle configurazioni di test nel rilevare i defect durante il processo di test.	
Test eseguiti	KPI correlato alla velocità del piano di esecuzione dei test. Tuttavia, questo KPI non offre una visione della qualità del codice.	
Densità di defect	Aiuta il team a identificare il numero di defect trovati in un software. Consente al team di decidere se il software è pronto per la release o se richiede più test. La densità di defect in un software è calcolata per migliaia di linee di codice.	Densità Defect = Numero di defect/ dimensione modulo
Defect Leakage	È utilizzato dai tester per revisionare l'efficienza del processo di testing, prima dell'UAT. Se alcuni defect non vengono evidenziati nella fase di test precedente possono esser rilevati dagli utenti, il che è noto defect Leakage o bug Leakage.	Defect Leakage = (Numero totale di defect rilevati in UAT/ Numero totale di defect rilevati prima dell'UAT) x 100
Defect Removal Efficiency	Defect Removal Efficiency (DRE) fornisce una misura dell'abilità dei team di sviluppo nel rimuovere i vari defect dal software prima della release o dell'implementazione. Calcolata durante e attraverso le fasi di test, DRE è misurato per tipo di test e indica l'efficienza del metodo di rimozione adottato dal team di test. È una misura indiretta della qualità e della performance del software.	DRE = Numero di defect risolti dal team di sviluppo/ numero totale di defect al momento della misura
Defect Category	Questo parametro offre un insight nei differenti attributi qualitativi del software, l'usability, la performance, la funzionalità, la stabilità e molto altro. La categoria dei defect è un attributo dei defect in relazione agli attributi qualitativi del prodotto software.	Defect Category =defect appartenenti ad una precisa categoria/numero totale di defect.
Defect Severity Index (DSI)	Offre un insight sulla qualità del prodotto sotto test e aiuta a valutare la qualità degli sforzi del team di test.	Defect Severity Index (DSI) = ∑ (Defect * Severity Level) / Numero totale di tefect

Difetti corretti al giorno	Valutando questo KPI si è in grado di tenere traccia del numero di difetti corretti su base giornaliera e degli sforzi investiti dal team per correggere tali difetti e problemi. Inoltre, consente loro di vedere lo stato di avanzamento del progetto e le attività di test	
Rapporto di successo della risoluzione dei difetti	Calcolando questo KPI, il team di tester del software può scoprire il numero di difetti risolti e riaperti. Se nessuno dei difetti viene riaperto, si ottiene il 100% di successo in termini di risoluzione.	Rapporto di successo risoluzione difetti = [(Numero totale di difetti risolti) - (Numero totale di difetti riaperti) / (Numero totale di difetti risolti)] x 100

(Software Testing Metrics & KPIs, n.d.)
Per alcuni indicatori quali il Defect Leakage, il calcolo risulta complesso a causa delle sovrapposizioni di alcune delle fasi di test previste da pianificazione.

3.2.5.3 Rilascio in Produzione

In fase di progettazione dei casi di test sono stati definiti i criteri di accettazione. I criteri di accettazione confermano che la qualità e la funzionalità del software sono stati raggiunti in vista del rilascio in Produzione.

- 100% dei casi di test UAT eseguiti ("passed", "failed" o "n / a")
- Totale difetti aperti: 0 difetti critici, 3% difetti principali. I principali difetti aperti non devono avere un impatto sul business. Nonostante tutti i difetti debbano essere risolti in generale, ci sono casi in cui nessuna correzione può essere consegnata o in cui la correzione non può essere eseguita prima di Go Live. Questi difetti devono discussi. per decidere se rinviare o rimandare una soluzione e documenterà le soluzioni alternative e i passaggi successivi necessari.

3.2.6 Il piano di test

I rapporti con il primo Vendor, vengono interrotti durante il mese di luglio 2019, periodo in cui sono in fase di finalizzazione i test di integrazione delle componenti IT e sono in corso le attività di System Test. La data di avvio degli UAT è invece, ancora da definire.
Come si è detto in precedenza per evitare il blocco delle attività e per sfruttare al meglio i tempi a disposizione si è deciso di continuare le attività di system test, utilizzando dei servizi Mockati. Ovvero, poiché i servizi del primo vendor in grado di fornire la funzionalità aggiuntiva non erano più disponibili, si è deciso di simulare la loro azione attraverso dei Mock e di continuare le attività di test, per verificare il funzionamento delle specificità di front end sia su Internet Banking e che su App.

Si sono quindi analizzati i casi di test precedentemente definiti per le attività di System Test e di UAT e l'esecuzione dei Casi di Test è stata suddivisa fra varie fasi:

- System Test con Servizi Mockati
- UAT e No Regression Test con Servizi Mockati
- Re-test UAT dei principali test già eseguiti, con i servizi reali
- UAT e No Regression Test delle funzionalità sviluppate in Wave 1
- System Test delle funzionalità sviluppate in Wave 2
- UAT e No Regression Test delle funzionalità sviluppate in Wave 1

A questi si aggiungono due fasi di Beta Testing in modalità Family &Friends.

Le attività di System Test sono affidate ad un fornitore esterno a cui sono affidati anche compiti di Defect Management sia per i defect rilevati in fase di System Test che in fase di UAT. Le attività di UAT sono invece svolte dalla Test Factory della banca per il canale Internet Banking e dalle aree del Business della banca per il canale App Mobile.
In basso è riportato il grafico che evidenzia il rate medio target di esecuzioni con esito positivo necessario per il completamento delle attività di UAT e No Regression Test entro i tempi prestabiliti, sia per il Canale App che per il Canale Internet Banking.

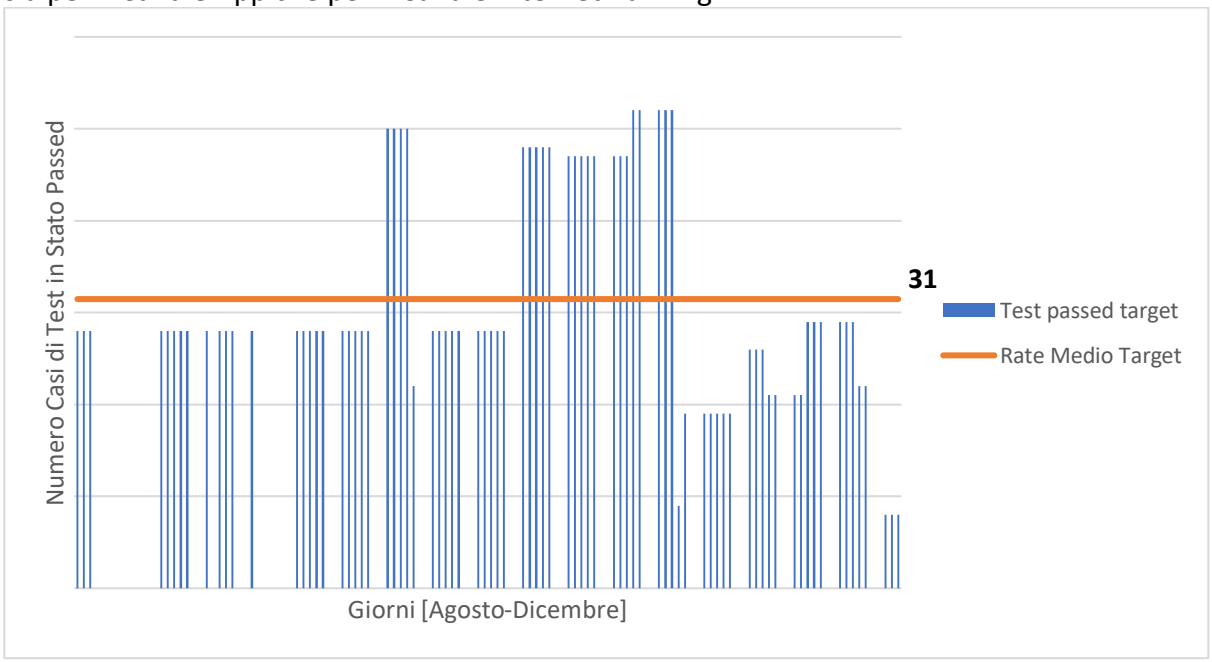


Figure 31 Distribuzione dei casi di test da eseguire giornalmente per la durata del progetto

Di seguito verranno analizzate alcune attività di test per il canale Internet Banking, le attività di test del canale app hanno caratteristiche che hanno portato ad una differente pianificazione, per ciò che riguarda le tempistiche di esecuzione, invece le modalità di esecuzione, le specificità riscontrate durante le varie attività di test e le modalità di gestione ricalcano esattamente quanto è avvenuto per il canale Internet Banking,

In effetti, durante l'esecuzione delle attività di test si è riscontrato un divario fra il rate di esecuzione dei test su App rispetto al rate di esecuzione del canale Internet Banking, ovvero i test su App erano caratterizzati da un rate di esecuzione più basso rispetto a quello che avviene sul canale Internet Banking.
Il rate inferiore dipende dalla maggiore complessità del test della Mobile App che a sua volta dipende da diversi fattori, fra cui il principale pare essere la necessità di testare diverse

configurazioni per diversi dispositivi. Per testare correttamente una Mobile App è necessario verificare:

- Il corretto funzionamento dei meccanismi di installazione e disinstallazione dell'applicazione, solitamente effettuato tramite test Smoke[12] e l'assenza di arresti anomali.
- Le prestazioni dell'applicazione nelle diverse reti 4G, 3G, Wi-Fi (Test di rete).
- La compatibilità con i vari dispositivi
- Il consumo di RAM e memoria

Tutto questo deve essere verificato ponendo particolare attenzione alla user experience, il cui impatto sul successo di un prodotto è ancora maggiore di quello che si ha per un prodotto computer based. (Mobile Application Testing vs. Web Application Testing, n.d.)

3.2.6.1 UAT con Servizi Mockati e Retest

Ad agosto sono state avviate le attività di UAT relativi al subset di test già eseguibili con servizi Mockati, con data attesa di completamento a fine settembre.
L'esecuzione dei Casi di test identificati per questa fase è stata completata a fine novembre. Per completare l'esecuzione dei casi di test entro fine settembre come da pianificazione i team di Test avrebbero dovuto sostenere un rate di esecuzione giornaliera pari a circa 17 Test passed al giorno, il rate sostenuto è invece inferiore del 67%. Il ritardo è dovuto a diversi fattori.
A fine settembre sono stati eseguiti il 55% dei casi di test di UAT già testabili con i servizi Mockati.
Nel meeting settimanale di aggiornamento sulle attività di Test dell'ultima settimana di settembre si è condivisa la necessità di un monitoraggio più attento delle attività di Test e la necessità di prendere adeguate misure per evitare ulteriori ritardi rispetto al ritardo sulle prime scadenze intermedio.
I driver di miglioramento individuati puntano ad aumentare il rate di esecuzione, incrementando le risorse disponibili per svolgere le attività di test, migliorare l'efficacia nell'attività di Data Preparation e ottimizzare ulteriormente i tempi di risoluzione dei defect segnalati durante gli UAT.

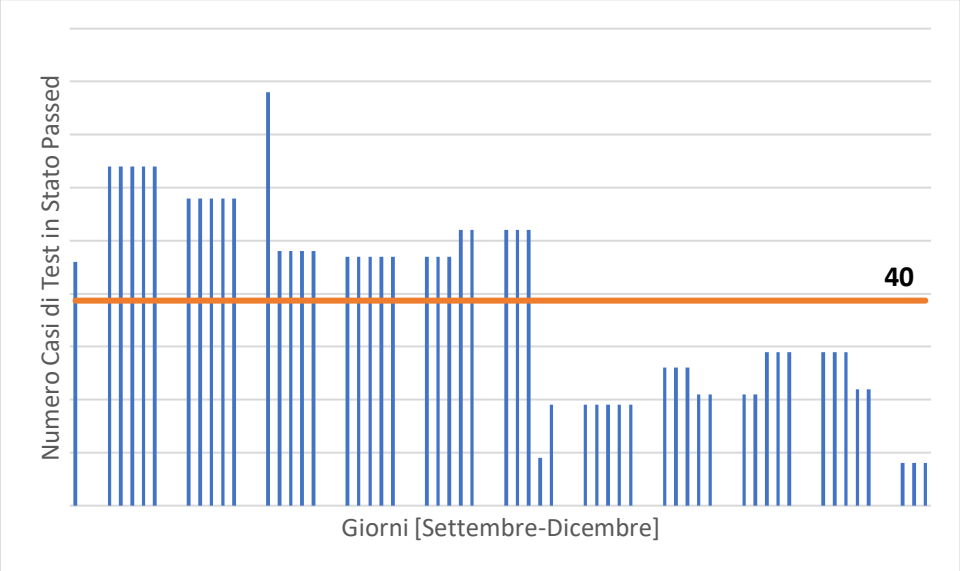


Figure 32 Distribuzione dei casi di test da eseguire giornalmente in seguito alla ri-pianificazione

[12] Gli Smoke Testing sono dei test preliminari al rilascio del software per verificare la presenza di eventuali gravi falle prima della release

A causa del ritardo sul primo insieme delle attività di test, il rate medio target di esecuzioni con esito positivo necessario per il completamento delle attività di UAT e No Regression Test entro i tempi prestabiliti è aumento del 33%.

I problemi identificati sono:

- risorse staffate insufficienti per sostenere i ritmi id esecuzione previsti
- risorse non formate adeguatamente per gestire test con servizi Mockati. Ovvero, i tester non sono formati per riconoscere comportamenti anomali della funzionalità legati alla presenza dei servizi Mockati, che non costituiscono difetti relativi al front end e alle interfacce che costituiscono l'oggetto di test in questa specifica fase.

Entrambe le problematiche derivano dalla ri-pianificazione delle attività. Come detto in precedenza, i team responsabili delle attività di Test sono team di tester della banca, impegnati su diverse progettualità, la ri-pianificazione delle attività ha portato ad un sovraccarico di lavoro per i team di tester. Se inizialmente il sovraccarico dei team di tester è determinato dall'essere impegnati su diverse progettualità in seguito, il sovraccarico viene a dipendere dalla sovrapposizione delle varie fasi di test della progettualità stessa. Sovraccarico che si riversa anche sui team di sviluppo impegnati nella risoluzione di una molteplicità di defect segnalati dalle diverse attività di test parallelamente in corso e per questo, a volte, duplicati.

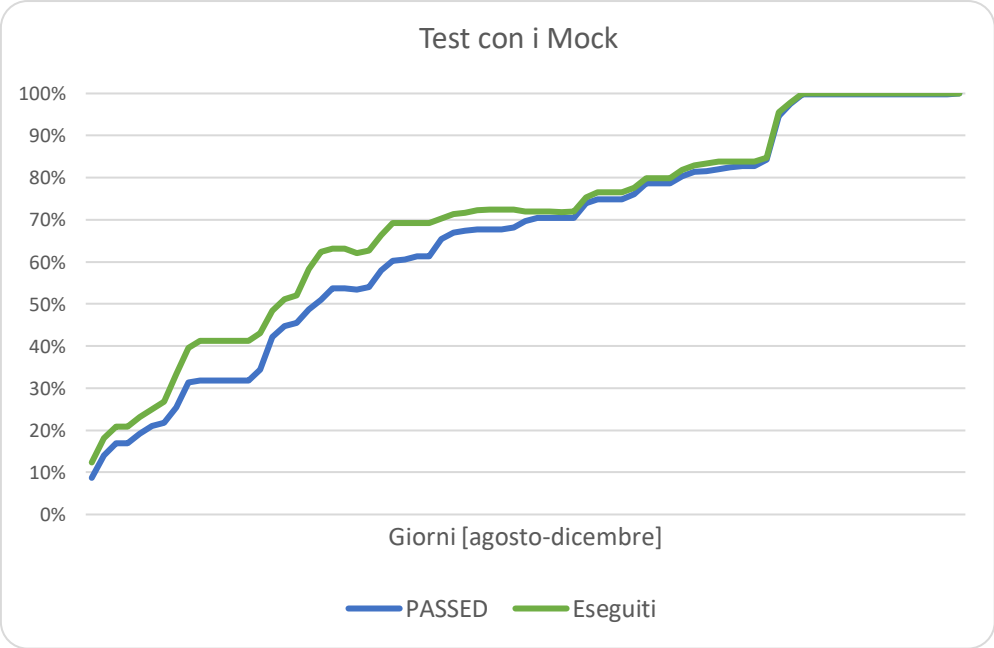


Figure 33 Andamento attività di test per i Test con i Mock

I defect segnalati sono spesso dovuti alla presenza dei Mock e per questo frequentemente sono respinti al tester (il 60% dei defect sono respinti al tester, una percentuale alta se si considera la media del 20% che caratterizza le altre fasi di UAT). I tester infatti segnalano comportamenti anomali dell'applicativo o comunque non rispondenti alle specifiche previste da requisito, questi comportamenti però si sono rilevati il più delle volte legati alla presenza dei Mock che simulano il comportamento della funzionalità.

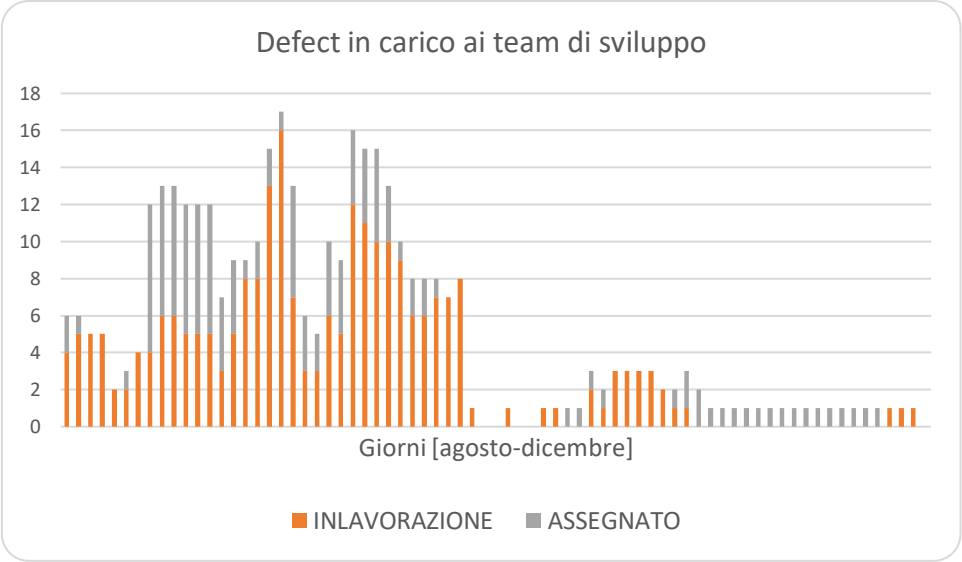


Figure 34 Andamento temporale dei defect in carico ai team di sviluppo per i test con i Mock

In seguito al rilascio dei servizi funzionanti sviluppati parte dei casi di test viene ritestata.

Le attività di Re-test sono caratterizzate da uno svolgimento differente, quasi opposto. Il rate di esecuzioni atteso per completare l'esecuzione dei casi di test come da pianificazione è quello di 3,5 test passed giornalieri, quello che si raggiunge è un rate pari a quasi il doppio.

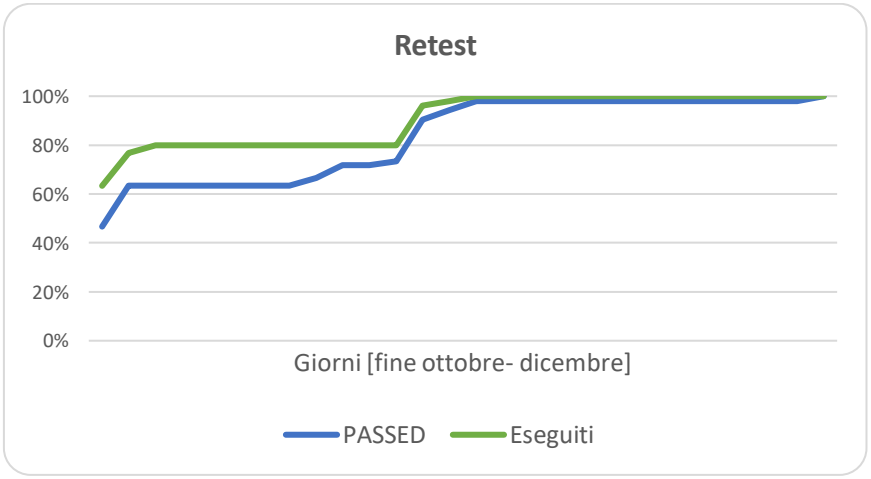


Figure 35 Andamento attività di test per i re-test

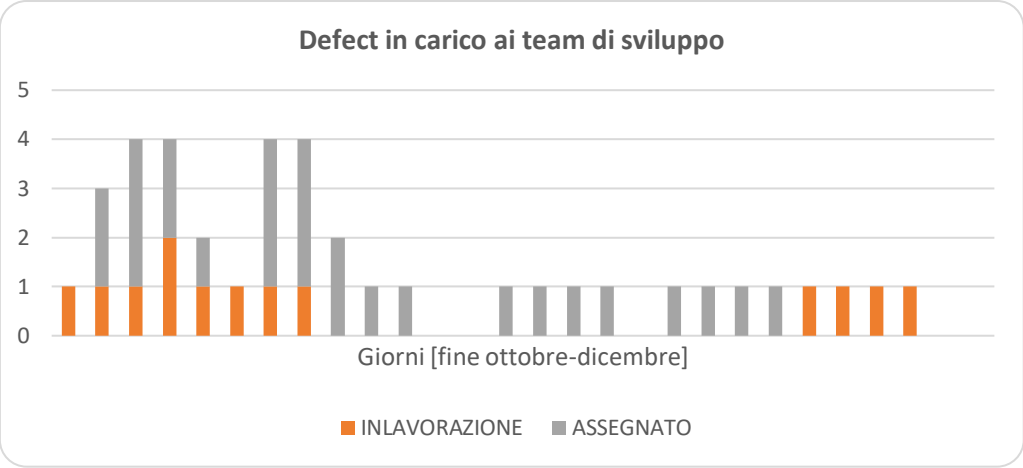


Figure 36 Andamento temporale dei defect in carico ai team di sviluppo per le attività di re-test

Il primo giorno sono state eseguiti il 60% dei casi di test previsti per questa fase e aperti i primi defect. Al rapido incremento della fase iniziale è seguito uno stallo delle attività della durata di 10 giorni, corrispondente al periodo in cui il numero di defect in carico ai team di sviluppo è massimo. In seguito, le attività sono riprese e si è arrivati celermente al completamento.
Se il basso rate di esecuzione nella fase intermedia delle attività di re-test era dovuto alle tempistiche necessarie per la risoluzione dei defect, quindi, diverso è il discorso per i test con servizi Mockati, dove il basso rate di esecuzione è spesso correlato alla mancanza di risorse.

Le attività di Re-test riguardano un sottoinsieme dei Casi di test già eseguiti con i test mockati, in particolare si tratta del test delle caratteristiche di front end, già testate che in qualche modo avrebbero potuto essere impattate dall'effettivo utilizzo dei servizi funzionanti.

Per verificare l'effettivo successo del processo di integrazione sono stati condotti No Regression Test.

3.2.6.2 UAT di Wave 1

In data 11/10 sono state avviate le attività di UAT di Wave 1 su Internet Banking per Chrome ed IE. Il rate di esecuzioni atteso per completare l'esecuzione dei casi di test come da pianificazione è quello di 6 test in stato passed al giorno, la media di esecuzione giornaliera ottenuta è invece pari al 76% in meno.

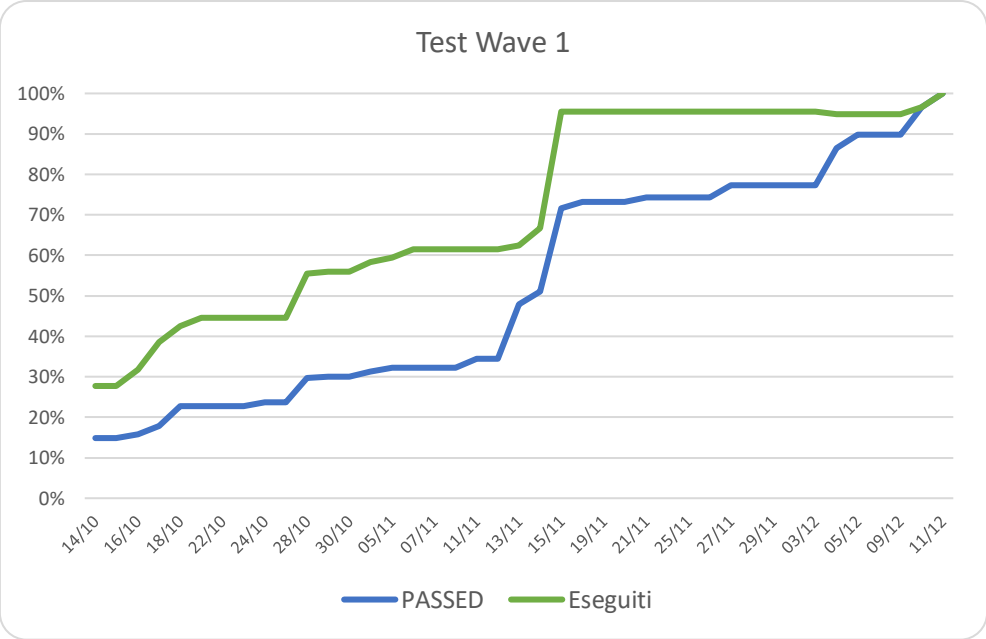


Figure 37 Andamento attività di test per Wave 1

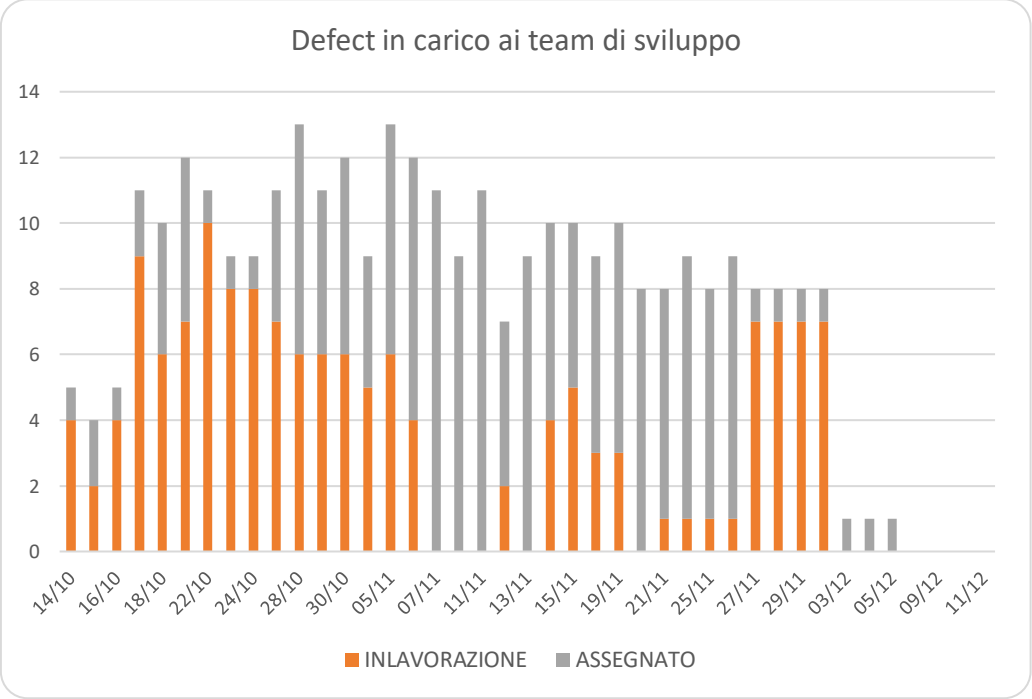


Figure 38 Andamento temporale dei defect in carico ai team di sviluppo per Wave 1

Le criticità principali della fase di test di Wave 1 sono legate all'ambiente PSD2. I test di Wave 1 avvengono infatti in ambiente di System Test, interagendo con l'ambiente di test Open Banking disegnato dalla normativa PSD2, ovvero in ambiente di Sandbox.

3.2.6.3 L'ambiente di Sandbox

Il 14 Giugno 2019 è il termine fissato dalla PSD2 per la disponibilità delle API di produzione funzionali da parte degli istituti finanziari e la prima data disponibile per i TPP per iniziare il processo di integrazione delle API. La Fintech Tink nella stessa data riporta che:
"*le API finora disponibili sono tutt'altro che pronte, mancano della qualità e della maturità di cui c'è necessità e offre un'anteprima che fa riflettere sulla condizione delle API di produzione che usciranno oggi*" (Tink , n.d.)
La Fintech infatti fornisce una descrizione della maturità delle API del panorama dell'Open Banking e della loro esperienza con le sandbox disponibili.
Il 14 Marzo 2019 gli Istituti Bancari e di credito avevano reso disponibili le proprie Sandbox. Le Sandbox sono ambienti di Test rispondenti ai requisiti imposti dalla normativa pensati affinché i TPP possano, fino al momento della disponibilità delle API di Produzione, testare le connessioni con le nuove interfacce delle banche e i customer journey dei propri applicativi, utilizzando dei dati fittizi che dovrebbero essere rappresentativi del comportamento delle API di Produzione. In questo modo dovrebbe esser facile passare dall'interazione con le API di Sandbox all'interazione con quelle di Produzione.
Il team di Tink che ha lo scopo di integrare le API di istituti bancari di 12 differenti mercati ha analizzato lo stato delle Sandbox di oltre 100 Istituti nel periodo fra il 14 Marzo e il 14 Giugno 2019 e dall'analisi ha elaborato sei criteri che definiscono una buona Sandbox:

1. Disponibilità e accessibilità. È il requisito base. L'accesso alle Sandbox deve essere garantito attraverso i Developer Portal degli Istituti Bancari, online form o delle API predisposte appositamente.
2. Documentazione. Il 14 Marzo gli Istituti bancari erano tenuti dalla normativa alla pubblicazione dei manuali per l'integrazione delle API PSD2.
3. Identificazione dei TPP. Per accedere alle Sandbox i TPP devono identificarsi attraverso i certificati eiDAS o attraverso certificati eiDAS di Test. Deve quindi essere prevista un'apposita modalità che consenta alle banche di autenticarsi.
4. Autenticazione e user experience. Una sandbox di qualità dovrebbe fornire un processo per consentire il test della modalità di SCA prescelta dall'istituto bancario.
5. Funzionalità. Alle TPP deve essere garantito l'accesso alle funzionalità e ai dati disponibili nell'Internet Banking e nell'App Mobile degli utenti finali
6. Testabilità. Dovrebbero essere offerti dati fittizi, account di test e casi di test per dare ai TPP un'indicazione realistica sul comportamento delle API in ambiente di Produzione.

Le verifiche condotte dalla Fintech hanno rivelato invece che alcune Sandbox non sembrano esistere, altre sono state affidate ad un provider esterno ma non vi sono indicazioni su quelle sia e altre ancora risultano ancora non disponibili. Le procedure di accesso inoltre, in molti casi richiedono tempistiche lunghe, anche di mesi. La documentazione offerta, presenta molte lacune e risulta spesso difficile da comprendere ed è di scarso supporto nel condurre le attività di test. Il processo di identificazione dei TPP attraverso i certificati eiDAS, non sempre implementato, differisce da banca in banca e questo non fa che allungare ulteriormente i tempi necessari per l'integrazione. Solo pochi casi consentono di testare effettivamente i passaggi di autenticazione dell'utente finale: le API di autenticazione a volte non sono presenti e altre non hanno un flusso definito. Il flusso di autenticazione utilizzato maggiormente è quello redirect, che offre una user experience di scarsa qualità. E anche quando l'autenticazione riesce, sono rari i casi in cui viene consentito il test delle funzionalità. Le Sandbox hanno inoltre pochi dati disponibili e spesso a diverse interrogazioni delle proprie API fanno corrispondere un'unica risposta.

Tutti questi fattori ostacolano le attività di test condotte dai TPP. Integrare queste API risulta tecnicamente impossibile ed è altrettanto costruire su queste un prodotto o un servizio in grado di portare valore agli utenti finali.

3.2.7 La gestione dei defect

Le attività di test Wave 1 rispetto ai test di UAT con i Mock, pur riguardanti lo stesso prodotto software, sono caratterizzati da una complessità maggiore.
Ciò che li differenzia è il fatto che i test con i servizi Mockati non testano il funzionamento del software nella sua interazione con l'ambiente PSD2 esterno, che è invece simulato dai Mock.

Diversa è la situazione nel caso dei test di Wave 1, dove la funzionalità si trova a dover interagire con l'ambiente esterno PSD2, anche questo un ambiente di Test, ovvero con le Sandbox.

Quindi in fase di Wave 1 vengono testate le funzionalità dell'applicativo che interagiscono con l'ambiente esterno, invece in fase di test con Mock vengono testate le caratteristiche proprie del prodotto, in particolar modo elementi di Front End, elementi grafici e legati alla user experience.
Questi elementi vengono poi ritestati in fase di Retest, una volta disponibili i servizi funzionanti, senza l'utilizzo dei Mock. In effetti le attività di Retest risultano facilmente gestibili e si rileva un rate di esecuzione delle attività sostenuto, così some quello di correzione dei defect, insieme ad un indice di difettosità globale relativamente basso, pari al 17%, se confrontato con le altre fasi di test.

La maggiore complessità delle attività si traduce in una maggiore difettosità ed in una maggiore complessità di gestione dei defect.

Table 5 Comparazione metriche di due fasi di test

Indici Test	UAT Servizi Mockati	UAT Wave 1
Tempo Medio Risoluzione Defect	17 giorni	26 giorni
Indice Riciclo	1,19	1,84
Rapporto di successo della risoluzione dei difetti	0,86	0,73
Indice di difettosità	20,95%	43,86%
Indice di Severità[13]	0,44	0,47
Percentuali Respinti	57%	96%
Percentuali Defect Bloccanti	56%	58%

Nella riprogrammazione delle attività di test non si è considerato che il rischio derivante da ambienti PSD2 instabili impattava non solo sulle attività stesse di test ma anche sulla loro gestione.

[13] *severità minima 1, massima 0*

La mancata definizione di un metodo di gestione appropriata per i casi di test ed i defect impattati da elementi esterni ha portato a continui rimbalzi fra i vari team di sviluppo per la correzione dei defect, prolungando così i tempi di esecuzione delle attività di test.
Infatti, la percentuale di respinti dei defect di Wave 1 è pari al 96% ed il tempo medio di correzione è di 25,5 giorni, entrambi i dati maggiori rispetto al test con i Mock.
La complessità è legata al test di una funzionalità che interagisce con un ambiente esterno, instabile e spesso non conforme agli standard previsti dalla normativa.

L'impossibilità di gestire la complessità, agendo sulla fonte costituita dall'ambiente esterno ha portato alla necessità di adottare un approccio adattivo: la correzione dei defect ha spesso portato a modifiche del software per la conformazione a specificità esterne.
Le continue modifiche richieste hanno reso necessario adottare un metodo di gestione più agile, più appropriato per la gestione di contesti turbolenti.
Tuttavia, come si è fatto notare precedentemente, la composizione del team di progetto non consente di adottare metodi agili nella totalità degli aspetti progettuali, in questo caso l'*agilità* è inserita all'interno di un framework strutturato.

Prima del cambio di vendor le incertezze dell'ambiente PSD2 erano affidate esclusivamente al vendor quindi, anche i defect derivanti da un ambiente PSD2 stabili sarebbero state immediatamente assegnate alla Fintech che avrebbe provveduto alla loro gestione.

Nella nuova pianificazione la gestione dell'incertezza legata alla PSD2 è affidata al nuovo vendor ma anche al team di progetto interno.
Le funzionalità di cui questi sono responsabili non sono divise in maniera netta ma esistono sovrapposizioni o comunque correlazioni che rendono complessa la gestione dei defect e l'attribuzione di un defect al giusto team di sviluppo.

Il ruolo di defect manager è affidato ad un fornitore esterno che lavora in una sede differente dal luogo di lavoro dei team di sviluppo ma anche da quello dei tester, a loro volta collocati in sedi differenti. La comunicazione avviene per lo più tramite e-mail ed on-line meeting ed è spesso frammentata.
In contesti di questo tipo, non è possibile adottare approcci agili e basati sul lavoro di team piccoli in grado di adattarsi alla variabilità dell'ambiente esterno ed in cui è garantita l'efficacia dei processi comunicativi. Pertanto, un'accurata documentazione delle attività, la definizione e la condivisione di procedure appropriate insieme all'utilizzo corretto di strumenti di test management e defect management, facilitano i processi comunicativi.

Si è deciso quindi di centralizzare le attività di defect manager e affidarle al team di PMO e di intensificare gli online meeting sull'andamento delle attività di test.

In fase di avanzamento delle attività, con l'aumentare delle richieste di modifiche si è poi deciso di creare un team cross funzionale composto da alcuni tester, sviluppatori interni e sviluppatori della fintech, insieme ad un rappresentante delle aree del business. Il team è responsabile della risoluzione dei defect correlati a fattori esterni alla progettualità e dipendenti dal contesto PSD2.
La task force ha proposto l'inserimento di una nuova terminologia per la classificazione e la gestione dei defect: Postponed.
Il termine è solitamente utilizzato con il significato di defect non ostacolante per il rilascio in produzione la cui risoluzione è posticipata alla release successiva.

Diverso è il significato nella nuova accezione. I defect dipendenti da ambiente PSD2 vengono analizzati dalla task force che fa un'analisi costi benefici della risoluzione tempestiva del defect. I costi sono calcolati in termini di ore uomo necessarie per la risoluzione, ma anche in termini di impatto sul prodotto delle modifiche richieste; i benefici, invece sono fatti dipendere dalla percezione di valore dell'utente finale in merito alla funzionalità che la risoluzione di un defect o un insieme di defect renderebbe disponibile. Da qui la necessità di inserire nel team un membro delle arre del business.
L'analisi può variare con l'andamento della progettualità, sia perché l'ambiente esterno è variabile, ma anche ad esempio per via di una maggiore disponibilità in termini di risorse per la risoluzione, per questo periodicamente la task force ha il compito di analizzare i defect Postponed ed eventualmente di richiederne la risoluzione.

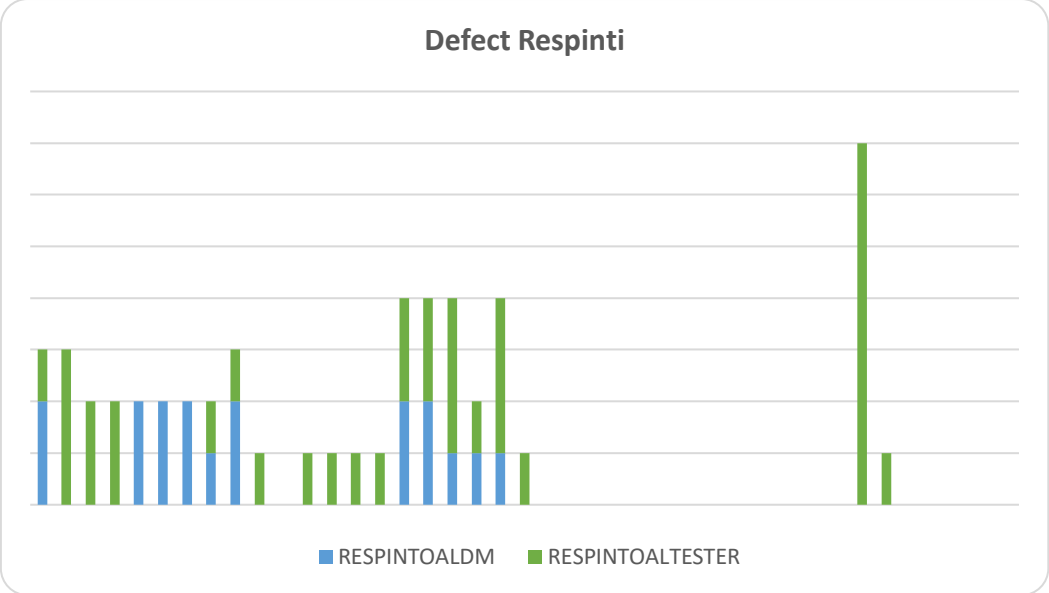


Figure 39 Andamento temporale defect respinti di Wave 1

L'effetto delle attività eseguite dalla task force ha condotto ad una riduzione dei rimbalzi dei defect prima dell'effettiva risoluzione.

3.2.8 Family & Friends di Wave 1

I test in modalità Family & Friends hanno avuto inizio in seguito al rilascio in Produzione di un perimetro iniziale di funzionalità. Si è quindi passati a testare dall'ambiente di System a quello di Produzione.
Le attività di beta testing sono molto utili poiché consentono di interagire con potenziali consumatori reali, ma allo stesso tempo si tratta di un'attività sfidante, difficile da gestire ed organizzare. Selezionare le risorse adatte infatti, può risultare time-consuming. Le difficoltà maggiori si riscontrano non solo in fase di recruiting di risorse per l'esecuzione dei test beta, ma anche nella formazione di queste in modo da riuscire ad effettuare test completi e raccogliere feedback in maniera continua e appropriata. (Should Employees, Friends, & Family Be Your Beta Testers?, n.d.)

Effettuare i Test in modalità Family & Friends consente facilita alcuni aspetti della gestione. Infatti, amici, familiari e dipendenti sono facili da reclutare per provare il prodotto, si tratta poi di risorse affidabili e meno costose.
In questo caso, gli utenti che hanno eseguito le attività di beta testing sono membri del team di progetto e famigliari stretti, scelti anche per preservare le informazioni sul progetto.
Il Family & Friends consente poi di implementare metodi di monitoraggio delle attività di test più strutturati. L'andamento dei Test di Family & Friends di Wave 1 è stato monitorato attraverso l'utilizzo di Google form inviati agli utenti, i defect invece sono stati censiti in un'apposita cartella creata nel Web Tool HPQC.

Prime indagini sullo stato dell'ambiente di Produzione si erano già svolte grazie al rilascio tecnico avvenuto a metà ottobre. Le prime indagini avevano rilevato delle difformità fra l'ambiente di Produzione e l'ambiente di Sandbox e una scarsa maturità delle API di Produzione, confermate dalle prime attività di Family & Friends di Wave 1.

Le caratteristiche dei test di Family & Friends ricalcano quelle delle attività di Wave 1 in termini di complessità rilevata e modalità di gestione.

3.2.8.1 Rilascio in Produzione

A seguito del rilascio in produzione delle funzionalità di Wave 1, sono stati attivati i test di UAT delle funzionalità di Wave 2.

Le funzionalità di Wave 2 hanno lo stesso principio funzionale di quelle di Wave 1 nel complesso, in aggiunta trattano alcune specificità relative all'ambiente PSD2.
Le attività di test di UAT di Wave 2, così come il relativo Family & Friends, sono state caratterizzate da analoga complessità rispetto a quanto accaduto per la Wave 1 ed analoghe modalità di gestione.

I ritardi sulle attività di Test non hanno impattato sulla data prevista per l'apertura alla clientela. Durante il meeting progettuale in cui è stata finalizzata la decisione di Go Live si è deciso di limitare ulteriormente il perimetro funzionale rispetto a quello determinato in acceleratore e procedere all'apertura alla Clientela limitando le funzionalità del prodotto.

Parallelamente all'apertura alla clientela sono in corso attività di Family & Friends sulle funzionalità non ancora rilasciate, perché impattate da defect Postponed.

A gennaio 2020 la funzionalità è stata aperta alla clientela sul Canale Internet Banking.

4 CONCLUSIONI

L'elaborato si propone di analizzare le attività svolte nell'ambito di un progetto di sviluppo di un prodotto software con il fine di comprendere i fattori e le motivazioni che hanno portato a preferire i metodi di gestione prescelti rispetto ad altri disponibili.

Il Progetto che costituisce l'oggetto dell'analisi riguarda lo sviluppo di un prodotto software di uno dei maggiori gruppi bancari italiani.

Il progetto nasce e si sviluppa nella sfera della PSD2. Lo scopo principale della Payment Service Directive è quello della tutela del consumatore accelerando la trasformazione del settore dei servizi finanziari da un universo product centric a universo consumer centric. La PSD2 vuole offrire al consumatore migliori standard in termini di sicurezza, una maggiore convenienza e allo stesso tempo aumentare la qualità e la varietà dei servizi offerti.

Per realizzare i propri obiettivi la normativa pone alle banche nuove sfide tecniche da affrontare che riguardano la sicurezza, l'implementazione di modalità di accesso ai propri sistemi, l'adattamento ad un sistema turbolento ed in continua evoluzione.

Le sfide poste alle banche non sono solo quelle tecniche. Abilitando l'ingresso di nuovi player nel settore e richiedendo la trasparenza dei modelli di pricing adottati dagli operatori finanziari, la normativa incrementa la competizione nel settore e spinge i prezzi verso il basso.

Se da un lato la PSD2 si presenta come una minaccia ai modelli di business tradizionali, dall'altra offre e abilita diverse opportunità di innovazione. La possibilità di operare come AISP e PISP non è infatti prerogativa esclusiva dei Third Party Provider; le stesse banche possono adottare i nuovi business model abilitati dalla normativa e offrire servizi e prodotti complementari a quelli offerti.

Al fine di rispondere a queste esigenze è necessario che le aziende adottino modelli di gestione dei processi in grado di farvi fronte. In particolare, per esser competitivi è necessario soddisfare due requisiti:

- ottenere time to market brevi
- soddisfare le esigenze dei consumatori

Si sono quindi analizzati diversi modelli di gestione dello sviluppo software a partire dai primi sviluppati ma ancora oggi utilizzati in determinati contesti, fino ai più recenti approcci Agile. Nell'analisi si evidenziano i vantaggi e gli svantaggi di ciascun modello ed in relazione ad essi suggerisce la tipologia di progetto per cui il modello si ritiene adatto.

Dall'analisi effettuata è evidente che non esiste un modello che possa essere applicato per tutte le tipologie di progetto. Ma la scelta del modello da adottare è una determinante per il successo di un progetto.

Una volta analizzata l'adattabilità di ciascun modello alle diverse progettualità, quindi, si è passati ad illustrare il progetto oggetto di studio, evidenziando le caratteristiche che hanno portato alle scelte adottate nella gestione del progetto.

L’esperienza del caso di studio proposta ha evidenziato come la scelta di un approccio ibrido sia quella che consente di gestire ciascuna fase progettuale in funzione della propria specificità con la metodologia più adatta, mitigando le conseguenze svantaggiose che ciascuna approccio comporta quando è implementato.

Il progetto oggetto del caso di studio è un progetto dove:

- il contesto è instabile e poco maturo
- i requisiti possono subire variazioni, anche frequenti
- le risorse sono disponibili e competenti: le attività di integrazione sono condotte da team esperti dei processi di integrazione
- l’organizzazione è gerarchizzata e strutturata
- operano diversi contributori, localizzati in differenti sedi: i team di sviluppo e test appartengono alla sezione dei Sistemi informativi della Banca e a 5 differenti fornitori, compresa la Fintech e che sono dislocati in diverse città del Nord Italia.
- il progetto software non richiedere tempistiche di esecuzione lunghe
- tecnologie e processi sono noti: i team quindi hanno una conoscenza approfondita dei processi di integrazione
- è richiesto un time to market breve.

La struttura organizzativa, costituita da team dislocati in varie sedi e la necessità di una documentazione adeguata ostacolano l’adozione di processi agili. Queste specificità sono infatti meglio gestire da approcci tradizionali.

Nella fase iniziale della pianificazione pertanto, le caratteristiche della progettualità conducono all’adozione di un approccio più tradizionale a livello macro. Sebbene la definizione del prodotto segua logiche agili in modalità MVP dettate dalla necessità di garantire time to market brevi e dalla poca maturità del contesto PSD2.

Per garantire time to market brevi e gestire l’incertezza si è cercato di implementare metodologie agili all’interno del framerwork tradizionale: per la definizione del requisito e l’analisi tecnica funzionale si sono adottate logiche agili e in un’ottica Rolling wave planning, le pianificazioni di dettaglio definite sono quindi quelle che guardano ad obiettivi di breve termine. Le pianificazioni di lungo periodo sono semplicemente accennate.

Nella fase iniziale della pianificazione quindi, si sono utilizzati approcci agili per rispondere all’esigenza di garantire time to market brevi e allo stesso tempo per consentire una miglior gestione delle incertezze. L’agilità richiesta da alcune specificità progettuali è stata però inserita in un framework più strutturato e tradizionale per rispondere ad altre caratteristiche della progettualità, dettate dalla struttura dei team di sviluppo, dalla necessità di documentazione, dalle competenze in merito ai processi di integrazione noti e dalla composizione e la localizzazione dei team di progetto.

A seguito del cambio Vendor le complessità introdotte con la nuova soluzione ad utilizzare logiche più agili.

Gli sviluppi interni della funzionalità determinano due elementi di complessità: il primo derivante dal fatto che la gestione dell'incertezza relativa alla PSD2 non è più affidata esclusivamente al vendor, ma deve essere gestita congiuntamente dal nuovo fornitore e dal team di sviluppo interno; il secondo relativi allo sviluppo congiunto di alcune funzionalità da parte di due team differenti con differenti approcci, localizzati in due differenti città italiane.

Il rischio maggiore identificato per la progettualità è quello relativo all'assunto, individuato nelle prime fasi progettuali, in fase di elaborazione del Project Charter, sulla disponibilità degli ambienti di test e produzione della PSD2 delle parti Terze, necessario per la validazione della funzionalità sviluppata. Il rischio correlato alla veridicità dell'assunto era gestito inizialmente dalla prima Fintech ingaggiata, ora la gestione deve esser condivisa fra il nuovo fornitore e il team di progetto interno.

Per una migliore gestione dell'incertezza si è deciso di adottare un approccio iterativo ed incrementale, dividendo la pianificazione in 2 wave e pianificando 3 differenti rilasci. A ciascuna wave è associatolo sviluppo di parte della funzionalità definita in MVP e per ciascuna wave è prevista una fase di Family & Friends.

Il cambio Vendor e la conseguente ri-pianificazione hanno avuto un forte impatto sulle attività di test, causandone il ritardo rispetto a quanto previsto da pianificazione.

Particolare attenzione quindi si è posta all'analisi della gestione delle attività di Test, quindi alle attività di Test Management e Defect Management che si sono rivelate le attività più critiche e time consuming. La criticità delle attività di test è legata in particolare alle specificità del contesto PSD2, che hanno richiesto un'attenta e specifica gestione al fine di poter rilasciare un prodotto software il più possibile privo di difetti o malfunzionamenti e rispondente alle aspettative della clientela.

In un contesto consumer centric come la PSD2 è fondamentale per rimanere competitivi realizzare prodotti in grado di soddisfare le sempre crescenti aspettative dei consumatori. Questo assume particolare rilevanza in questo caso, dove il prodotto è rilasciato da una banca, parte di uno dei maggiori gruppi bancari italiani che ha quindi una brand reputation da difendere ed è per questo che garantire elevati standard qualitativi del prodotto è di cruciale importanza.

In particolare, fra le attività di test in quest'ottica assumono particolare rilevanza le attività di UAT, la cui complessità aumenta nel momento in cui i tester si trovano a dover lavorare in ambiente PSD2.

La ri-pianificazione delle attività ha impattato sia le attività di esecuzione di UAT di Wave 1 che le attività di UAT con i Servizi Mockati. Gli impatti sono correlati a differenti fattori, ciascuna con specifiche conseguenze:

- Sovraccarico dei team di Test e di sviluppo
- L'ambiente PSD2.

Il primo fattore ha determinato un ritardo nell'esecuzione dei Test con i Mock, ritardo che si è ripercosso sulle successive attività di Test.

Diversa è la situazione nel caso dei test di Wave 1, dove la funzionalità si trova a dover interagire con l'ambiente esterno PSD2, anche questo un ambiente di Test, ovvero con le Sandbox. La maggiore complessità è legata al test di una funzionalità che interagisce con un ambiente esterno, instabile e spesso non conforme agli standard previsti dalla normativa.

Se nel caso dei test con Mock, l'inadeguatezza delle risorse staffate aveva condotto al ritardo sui tempi di esecuzione, la mancata definizione di un metodo di gestione appropriata per i casi di test ed i defect impattati da elementi esterni ha portato a continui rimbalzi fra i vari team di sviluppo per la correzione dei defect, prolungando così i tempi di esecuzione delle attività di test. Infatti, nella nuova soluzione adottata in seguito al cambio vendor, il nuovo fornitore ed il team interno sono responsabili entrambi dello sviluppo della funzionalità e della gestione dell'incertezza relativa alla PSD2. Tutto ciò aumenta il grado di complessità nell'attribuzione di un defect al giusto team di sviluppo

La variabilità e l'incertezza dell'ambiente esterno richiederebbero un approccio di gestione più agile che è ostacolata dall'organizzazione. Ancora una volta, si decide per un compromesso fra i diversi approcci agendo quindi sia su documentazione e procedure e adeguandole alla nuova situazione ma anche adottando approcci basati sul lavoro di team piccoli e cross funzionali, tutto svolto con lo scopo di migliorare l'efficacia dei processi comunicativi.

I defect dipendenti da ambiente PSD2 vengono analizzati da una task force che fa un'analisi costi benefici della risoluzione tempestiva del defect, che ne definisce l'attribuzione e le tempistiche di risoluzione.

Le modalità di gestione adottate in ambiente di test sono state replicate per la gestione dei defect rilevati nella fase di test in Produzione, ovvero in fase di Family & Friends di Wave e Wave 2.

Il caso di studio evidenzia quindi, come il metodo di gestione più adatta dipenda dal contesto di riferimento, interno ed esterno, dalle caratteristiche della progettualità e che ciascuna fase possa essere gestita con il metodo e l'approccio che più si adatta alle proprie specificità.

5 BIBLIOGRAFIA

(s.d.). Tratto da TutorialsPoint: https://www.tutorialspoint.com/software_testing_dictionary/test_bed.htm

12 Principles Behind the Agile Manifesto. (s.d.). Tratto da https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

7 Software Testing Principles: Learn with Examples. (s.d.). Tratto da guru99.

Agile Manifesto. (s.d.). Tratto da https://www.agilealliance.org/agile101/the-agile-manifesto/

Agile Tetsing. (s.d.). Tratto da https://www.guru99.com/agile-testing-a-beginner-s-guide.html

Ahmed, A. (2011). *Software Project Management A Process-Driven Approach.*

Alessio Botta, N. D. (s.d.). Tratto da https://www.mckinsey.com/industries/financial-services/our-insights/psd2-taking-advantage-of-open-banking-disruption

Bonzer Technology. (2016). *Lean Software Development and Minimum Viable Product*. Tratto da https://blog.bonzertech.com/lean-software-development-and-minimum-viable-product-mvp/

Cockburn, A. (2000). *Selecting a Project's Methodology*. Tratto da https://courses.cs.ut.ee/MTAT.03.243/2015_spring/uploads/Main/cockburn.pdf

Comai, A. (2013). *Metodologie di sviluppo software: essere pragmatici*. Tratto da https://www.zerounoweb.it/software/sviluppo-software/metodologie-di-sviluppo-software-essere-pragmatici/

Deepty, A. e. (2013). *A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios.*

Deloitte. (2019). *Open Banking e API Economy.*

Deloitte. (s.d.). *The Bank as a Platform.* Tratto da https://www2.deloitte.com/it/it/pages/financial-services/articles/bankasaplatform-italy-financialservices.html

Fabrick. (s.d.). *PSD2 Gateway: Operational Guidelines for AISs and PISs.* Tratto da https://docs.fabrick.com/psd2/manuals/OperationalGuidelinesForTPP.pdf

Fido Alliance. (2018). *Fido for PSD2.* Tratto da https://fidoalliance.org/wp-content/uploads/FIDO-PSD2_Customer_Journey_White_Paper.pdf

FIDO and PSD2: Solving the Strong Customer Authentication Challenge in Europe. (2018). Tratto da https://fidoalliance.org/fido-and-psd2-solving-the-strong-customer-authentication-challenge-in-europe/

Govardhan2, N. M. (2010). A Comparison Between Five Models Of Software Engineering. *IJCSI International Journal of Computer Science Issues*.

IBM. (2019). *Beyond PSD2.* Tratto da https://www.ibm.com/blogs/security-identity-access/wp-content/uploads/2019/01/Beyond-PSD2_Paper.pdf

Integration Testing. (s.d.). Tratto da https://www.guru99.com/integration-testing.html

Intesa San Paolo. (s.d.). *Ciclo di Vita del Demand Management.* Tratto da https://www.academia.edu/36678611/Ciclo_di_vita_del_Demand_Management_in_Intesa_Sanpaolo_Le_fasi_del_Demand

KPMG Advisory. (2018, Settembre). *Le sfide della direttiva europea PSD2.* Tratto da https://home.kpmg/it/it/home/insights/2018/09/le-sfide-della-direttiva-europea-psd2.html

Krishnan, E. W. (1997). A Model Based Framework to overlap Product Development Activity. *Management Science*.

Larman, C. (2004). *Agile and Iterative Development: A Manager's Guide.*

Laura Brodsky, C. I. (2018, Agosto). *McKinsey & Company*. Tratto da https://www.mckinsey.com/industries/financial-services/our-insights/open-bankings-next-wave-perspectives-from-three-fintech-ceos

Marcoli, C. (2018, Agosto). *IBM Developer*. Tratto da https://developer.ibm.com/apiconnect/2018/08/21/psd2_architecture/

Mary Poppendieck, T. P. (2003). *Lean Software Development: An Agile Toolkit.* Tratto da https://en.wikipedia.org/wiki/Lean_software_development

Mobile Application Testing vs. Web Application Testing. (s.d.). Tratto da https://www.nexsoftsys.com/articles/difference-between-mobile-web-software-testing.html

PSD2 EIDAS test Certificates (QWAC and QSeal). (s.d.). Tratto da https://www.eadtrust.net/en/2019/07/psd2-eidas-test-certificates-qwac-and-qseal/

PwC Italia. (s.d.). Tratto da https://www.pwc.com/it/it/industries/banking-capital-markets/psd2.html

Rational . (s.d.). *Rational Unified Process.* Tratto da https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Red Hat. (s.d.). *I vantaggi delle interfacce di programmazione delle applicazioni.* Tratto da https://www.redhat.com/it/topics/api/what-are-application-programming-interfaces

Royce, W. (1970). *Managing the Development of Large Software Systems.*

Should Employees, Friends, & Family Be Your Beta Testers? (s.d.). Tratto da https://www.centercode.com/blog/2014/05/should-employees-friends-family-be-your-beta-testers

Software Testing Metrics & KPIs. (s.d.). Tratto da https://www.nexsoftsys.com/articles/difference-between-mobile-web-software-testing.html

Sutherland, K. S. (2017). *La Guida a Scrum.* Tratto da https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Italian.pdf

System Testing. (s.d.). Tratto da https://www.guru99.com/system-testing.html

Tink . (s.d.). *The sobering September preview: banks' PSD2 APIs far from ready*. Tratto da https://blog.tink.com/blog/2019/06/14/psd2-updated-sandbox

Tuomas Neonen, A. T. (s.d.). *PSD2 Consumer Solution.* Tratto da https://www.nets.eu/solutions/Consumer%20Management%20Services-OLD/Documents/White%20paper_PSD2_Consumer_Solution.pdf

Unit Testing. (s.d.). Tratto da https://www.guru99.com/unit-testing-guide.html

User Acceptance Testing. (s.d.). Tratto da https://www.guru99.com/user-acceptance-testing.html

What is Agile? (s.d.). Tratto da https://www.agilealliance.org/agile101/

Ylenia Bezza, F. G. (2018, Luglio). *La rivoluzione digitale nel settore bancario.* Tratto da https://home.kpmg/it/it/home/insights/2018/07/digital-banking-2018.html

6 LISTA FIGURE E TABELLE