POLITECNICO DI TORINO

Collegio di Ingegneria Gestionale

Corso di Laurea Magistrale in Ingegneria Gestionale

PERCORSO SERVIZI

Tesi di Laurea di II livello

Modello ITIL e DevOps a confronto:

caso di studio di una azienda leader nel settore dei ricambi in ambito automotive

Relatore accademico:

Prof. Alberto De Marco

Candidato:

Giulia Mazzoni

Anno Accademico 2020-2021

Elenco delle figure

Sommario

INTRODUZIONE

Un modello di Governance funzionale riveste un'importanza strategica all'interno del contesto aziendale poiché, un sistema efficiente e ben progettato, garantisce una migliore gestione aziendale con ricadute positive sulla produttività e, quindi, sull'operato o business dell'azienda stessa.

Attualmente tutte le realtà aziendali più avanzate hanno messo in piedi un sistema di Governance più o meno complesso in base a diversi fattori ma, sempre di più, è necessario essere al passo con i tempi per poter essere competitivi sul mercato nel minor tempo possibile.

Per poter arricchire le conoscenze in questo ambito e vederle applicate in un progetto di servizi IT, è stato richiesto supporto a Techedge S.p.A., una società di consulenza specializzata nell'Information Technology, in grado di creare strumenti che rendono possibile l'integrazione tra diversi sistemi gestionali utilizzati da un'organizzazione e tra i propri fornitori.

Grazie a Techedge S.p.A. è stato possibile entrare in contatto con uno dei suoi principali clienti: una azienda leader nel settore dei ricambi automotive che utilizza per i suoi magazzini, come ERP, un SAP standard che, nel corso del tempo, è stato customizzato per poter soddisfare tutte le esigenze dei diversi stakeholders coinvolti.

Siccome il sistema in questione è molto complesso, per poter sopravvivere generando il valore atteso, necessita di un sistema di Governance IT altrettanto evoluto. L'obiettivo della tesi è quello di andare ad analizzare il sistema attualmente in funzione e capire se l'implementazione di un modello innovativo, come quello DevOps, sia applicabile e/o vantaggioso rispetto al modello ITIL V4 su cui si basa attualmente.

Nel primo capitolo verrà illustrato nel dettaglio il modello ITIL V4 e, a seguire, nel capitolo successivo verrà mostrato il modello DevOps.

Nel terzo capitolo si presenterà l'azienda Techedge S.p.A., la sua storia e la sua organizzazione.

Il capitolo 4 comprenderà una analisi del modello di Governance attuale e, in particolare, focalizzerà l'attenzione sui processi implementati sul principale sistema di trouble shooting utilizzato dal cliente.

Nel quinto capitolo sarà illustrata una possibile implementazione del modello DevOps, focalizzando l'attenzione soprattutto sul concetto di Smart Team e sui benefici che esso comporta.

Infine, si discuteranno i pro e i contro di questa implementazione sia da un punto di vista qualitativo che quantitativo, i possibili passi futuri e le conclusioni del candidato al termine del suo percorso di tesi e tirocinio.

1. IL MODELLO ITIL 4

1.1 Introduzione al modello ITIL 4

Figura 1 Parole chiave Modello ITIL 4

A partire dagli anni ’80, assieme alla diffusione delle grandi aziende e l’avvento delle multinazionali, si è vista la creazione di una nuova dipendenza tra business e tecnologia che ha portato alla propagazione di nuovi ed eterogenei servizi che hanno rivoluzionato l’intero sistema di gestione e coordinamento del flusso di lavoro.

Queste nuove tecnologie, seppur abbiano avuto la capacità di snellire e ottimizzare il lavoro quotidiano, hanno dovuto necessariamente portare il personale ad acquisire le conoscenze per poterle utilizzare (il service management) e gli strumenti per trattare più velocemente i problemi che da esse derivavano (l’IT Help desk).

L’IT service management (ITSM) è un concetto che permette ad una organizzazione di massimizzare il valore aziendale tramite l’utilizzo della “tecnologia dell’informazione”; esso posiziona i servizi IT come il mezzo chiave per fornire e ottenere valore ma, per assicurarne una qualità sostenibile, è stabilito un insieme di pratiche, o processi, che costituiscono un sistema di gestione dei servizi.

Un Service Manager professionale e responsabile è colui che porta questa qualità al servizio del business; in particolare egli deve definire il servizio offerto, gestire gli accordi sul livello di servizio (Service Level Agreement) e assicurare che i servizi soddisfino i requisiti e le esigenze espresse dal cliente.

Per comprendere cosa sia la gestione dei servizi però è necessario comprendere fin dal principio cosa sia un servizio: “Un servizio è l’output di una organizzazione quando, necessariamente, almeno una attività è svolta tra l’organizzazione e il cliente”. (Definizione ISO 900:2015).

La definizione ITIL ufficiale contenuta all'interno del volume Service Design, invece, definisce il servizio come “un mezzo per fornire valore ai clienti, agevolando i risultati che si intende ottenere, e senza l’assunzione di costi e rischi specifici.” (Office of Government Commerce, 2011)

Si possono individuare quattro prospettive ("4P") o attributi per spiegare il concetto di ITSM:

- Prospettiva Partner / Fornitori: prende in considerazione l'importanza dei partner e i rapporti con i fornitori esterni (e di come gli stessi contribuiscono al Service Delivery);
- Prospettiva delle Persone: definisce gli aspetti relativi alle persone coinvolte (personale IT, clienti e altre parti interessate);
- Prospettiva Tecnologica / di Prodotto: tiene conto dei servizi IT, hardware e software, bilanci e strumenti;
- Prospettiva di Processo: descrive il processo di consegna di un servizio analizzando tutte le fasi che intercorrono tra la domanda e l'offerta.

1.2 Il framework ITIL

Il framework ITIL (Information Technology Infrastructure Library) nasce in Inghilterra negli anni ’80 sotto la richiesta del Governo Britannico che, ritenendo la qualità dei servizi IT scadente, commissionò all’ente governativo OGC (Office of Government Commerce) la stesura di alcune linee guida standard e comuni per l’uso efficiente ed efficace delle risorse IT.

Durante gli anni ottanta, la diffusione di ITIL rimase limitata alla Gran Bretagna; il boom di utilizzo si ebbe circa un decennio dopo quando, dalla metà degli anni novanta, molte aziende iniziano ad utilizzarlo.

Naturalmente, questa prima versione era ampiamente differente dall’ITIL che è noto oggi infatti, nel corso degli anni, il framework è stato adottato da numerose aziende, si è evoluto e si è orientato sempre di più verso le esigenze del cliente ricercando un continuo miglioramento. La seconda versione di ITIL uscì nel 2001 mentre ITIL V3, il predecessore del modello attuale, venne pubblicato nel 2007; quest’ultimo comprende

cinque testi principali: Service Strategy, Service Design, Service Transition, Service Operation e Continual Service Improvement.

Infine, l'ultima versione, ITIL V4, è stata pubblicata nel 2019 e si concentra maggiormente sulla gestione dei servizi, che vanno dalla domanda alla creazione di valore, aiutando le aziende ad affrontare le novità dell'era tecnologica dei servizi digitali.

L'ultima versione di ITIL, rispetto alla precedente, non introduce ulteriori idee riguardanti la gestione del servizio, ma bensì l'introduzione di 34 pratiche che permettono ai professionisti del settore di comprendere meglio i principi e i concetti fondamentali come "valore" e "risultati". ITIL V4, inoltre, fornisce anche consigli per l'integrazione con altri framework e metodologie come DevOps, Lean e Agile.

L'adozione dell'ITIL è in grado di offrire agli utenti una vasta gamma di vantaggi che includono:

- riduzione dei costi;
- miglioramento della soddisfazione del cliente attraverso un approccio più professionale alla prestazione di servizi;
- miglioramento della produttività;
- migliore utilizzazione delle competenze ed esperienze;
- migliore fornitura di servizi di terze parti.

I componenti principali del framework ITIL 4, analizzati nel dettaglio successivamente, sono:

1.2.1 Il modello a quattro dimensioni

1.2.2 Il Service Value System (SVS)

1.2.1 Il modello a quattro dimensioni

Le quattro dimensioni non sono indipendenti ma sono sinergiche: se si pone poca attenzione anche soltanto su una singola dimensione il valore fornito dall'organizzazione sarà limitato per i suoi stakeholders.

Ci sono 6 fattori ambientali che possono influenzare positivamente o negativamente le 4 dimensioni e che, di conseguenza, devono essere considerati in ottica di gestione del rischio. I fattori in questione sono:

1) Politico: un cambiamento di leadership all'interno di un'organizzazione, o nella località in cui l'organizzazione opera, può portare a sentimenti positivi o negativi nei confronti dei servizi erogati; ad esempio, un governo può andare al potere con un approccio protezionistico o di apertura delle frontiere che può avere un impatto sulla quota di mercato di un'azienda.
2) Economico: il cambiamento dei tassi d'interesse o dei prezzi delle public utilities può comportare che i servizi di un'organizzazione siano sopravvalutati o non redditizi.
3) Sociale: le preferenze e le percezioni della gente cambiano nel tempo. Molte organizzazioni hanno diverse tipologie di stakeholders che appartengono ad un'ampia gamma di età e gruppi demografici; per questo motivo può essere difficile trovare un approccio unico per gestire il servizio.
4) Tecnologico: il modo in cui i servizi vengono creati ed erogati è enormemente influenzato dalla tecnologia; alcuni esempi recenti sono AI, big data o criptovalute.
5) Legale: il GDPR ha portato a cambiamenti significativi nel modo in cui i fornitori di servizi gestiscono i dati degli utenti dal punto di vista della privacy.
6) Ambientale: Il cambiamento climatico sta influenzando il modo in cui le organizzazioni vedono i loro servizi e la fornitura di servizi; i clienti stanno diventando propensi ad acquistare servizi da aziende che sono viste come rispettose dell'ambiente.

L'ITIL V4 definisce 4 punti di vista che devono essere presi in considerazione nella gestione dei servizi:

• Organizations and people

• Information and tecnology

• Partners and suppliers

• Value stream and processes

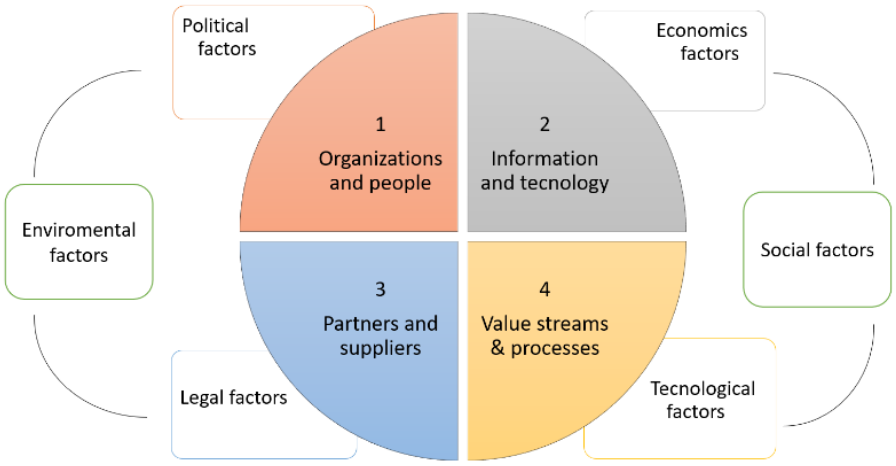


Figura 2 Il modello a 4 dimensioni

Tutte queste dimensioni devono essere considerate nel loro insieme, sarebbe infatti sbagliato concentrarsi più su un'attività piuttosto che su un'altra e si creerebbero maggiori inefficienze. Infatti, se non si trattassero correttamente tutte e quattro le dimensioni, i servizi potrebbero essere non consegnabili o non conformi alle aspettative di qualità/efficienza.

Si analizzino nel dettaglio le 4 dimensioni:

1) Organizations & people

Questa dimensione definisce tutti quegli aspetti legati principalmente alle persone che devono essere considerati quando si gestisce un servizio; le persone includono impiegati, manager, dirigenti, clienti, dipendenti dei fornitori e chiunque altro sia coinvolto nella creazione o nel consumo del servizio.

La complessità delle organizzazioni sta crescendo rapidamente e, per questo motivo, è importante conoscere e favorire, per quanto possibile, una struttura organizzativa e gestionale sempre più efficiente rispettando i ruoli, le responsabilità e i vari sistemi di

autorità e di comunicazione. Ogni individuo all'interno dell'organizzazione dovrebbe avere una chiara visione complessiva del suo contributo alla creazione di valore aggiunto per l'intera organizzazione, per i suoi clienti e per tutte le parti interessate.

2) Information & technology

Si riferisce ad aspetti informatici e tecnologici: ci interessa capire come gli strumenti IT supportano un servizio e come questi strumenti possono aiutare a gestire il portafoglio dei servizi.

Gli strumenti di collaborazione e i social media hanno aumentato il potenziale per la co-creazione di servizi e permettono una maggiore consapevolezza dei bisogni, delle esperienze e dei sentimenti dei clienti mentre i servizi cloud hanno reso più facile per le organizzazioni distribuire rapidamente, scalare o smantellare i servizi in risposta a esigenze sempre mutevoli.

È necessario chiedersi: "Quali informazioni servono per creare, cambiare o fornire valore?" oppure "Quali sono gli input e output di ogni fase del flusso di valore?"

Altri aspetti da considerare sono le competenze e la sicurezza: si nota che spesso è difficile che l'impresa possieda le giuste persone per costruire, mantenere e proteggere una nuova tecnologia. Per esempio, le organizzazioni che lanciano nuovi servizi informativi basati su architetture di big data, hanno difficoltà a trovare le persone "giuste" per supportare la tecnologia adottata.

Infine, siccome la sicurezza è diventata fondamentale nel modo in cui solo gestite le informazioni e la tecnologia, gli aspetti di garanzia dei servizi (disponibilità, capacità, sicurezza e continuità) sono sempre più determinati dall'utilizzo della tecnologia. Si deve sempre verificare che la tecnologia non crei problemi di conformità normativa e che quindi rispetti le policy aziendali e le normative GDPR.

3) Partners & Suppliers

Ogni organizzazione è sia fornitore che consumatore di servizi, tuttavia l'ampiezza con cui le imprese integrano i fornitori nelle loro catene del valore varia a seconda delle capacità interne, dei sourcing bias e dei requisiti normativi.

La relazione con i partner e i fornitori è solitamente definita in base a ciò di cui l'organizzazione ha bisogno: alcuni fornitori sono caratterizzati come *strategici* a causa del loro impatto critico sui servizi (in questi casi, il rapporto dovrebbe essere gestito ad un alto livello di leadership), altri invece sono facilmente sostituibili poiché operano a livello di commodity. L'organizzazione, e con lei anche i suoi partner e fornitori, devono avere una comprensione comune di come i loro sforzi di collaborazione forniscono valore attraverso i risultati, e cosa ci si aspetta da entrambe le parti riguardo alla creazione di valore.

Uno dei modi in cui le organizzazioni stanno gestendo le loro interazioni con partner e fornitori è attraverso una struttura chiamata SIAM (Service Integration and Management), che comporta l'utilizzo di un integratore per attuare e gestire processi comuni e coordinati tra più soggetti: tale struttura funge la protezione nei confronti del cliente finale che vede solo una singola organizzazione di fornitura di servizi, piuttosto che più fornitori. L'integrazione dei servizi può essere gestita dall'organizzazione o esternalizzata a un partner o a una terza parte indipendente, a seconda delle esigenze dell'organizzazione.

I fattori che potrebbero influenzare la strategia di un'organizzazione sono:

- Focus strategico: alcune organizzazioni potrebbero concentrarsi maggiormente nelle loro competenze core ed esternalizzare a terzi funzioni non principali. Altre, invece, potrebbero preferire rimanere autonome mantenendo il pieno controllo di tutte le funzioni importanti;
- Cultura aziendale: alcune organizzazioni hanno preferenze per un approccio rispetto che ad un altro, legate ad una cultura aziendale di lunga data e difficile da cambiare senza ragioni convincenti;

- Scarsità delle risorse: se una risorsa richiesta è carente all'interno dell'organizzazione, potrebbe essere complesso per un fornitore di servizi acquisire ciò che è necessario senza coinvolgere un fornitore esterno;
- Preoccupazioni sui costi: talvolta il fornitore di servizi ritiene che, per soddisfare un particolare requisito, sia più economico rivolgersi ad un fornitore esterno (in altri casi invece è valido il viceversa);
- Competenza in materia: il fornitore di servizi può ritenere che sia meno rischioso utilizzare un fornitore che ha già esperienza in un'area richiesta piuttosto che cercare di sviluppare quella specifica competenza internamente;
- Vincoli esterni: possono influire regolamentazione e politica del governo, codici di condotta del settore e vincoli politici o legali.

4) Value Streams & Processes

Questa dimensione serve per descrivere il modo con cui il valore rilasciato con un servizio può creare un fattore positivo/negativo sull'intera organizzazione. In altri termini, è necessario descrivere il valore rilasciato con ogni servizio per poter capire se è allineato con le richieste aziendali.

Questa dimensione definisce le attività, i flussi di lavoro, i controlli e le procedure necessarie per raggiungere gli obiettivi concordati. Ciò che conta, nella gestione dei servizi, è che un'organizzazione stabilisca un modello operativo che organizza efficacemente le attività chiave necessarie per gestire prodotti e servizi (solitamente sono dettagliati delle procedure, che delineano chi è coinvolto nel processo e in cosa consiste la sua mansione).

Per Value Streams si intende una serie di passi che un'organizzazione intraprende per creare e fornire prodotti e servizi ai consumatori: strutturare i portafogli di prodotti e servizi dell'organizzazione intorno ai flussi di valore permette alle aziende di avere un quadro chiaro di ciò che fornisce e in che modo. Mappando i suoi flussi di valore, un'organizzazione può identificare cosa è critico, cosa introduce sprechi e cosa può essere migliorato.

Per processi, invece, si intende una serie di attività correlate o interagenti che trasformano gli input in output. I processi definiscono la sequenza delle azioni e le loro dipendenze, così come descrivono ciò che viene fatto per realizzare un obiettivo. I processi sono sostenuti da politiche e possono essere ulteriormente suddivisi attraverso procedure che delineano cosa viene fatto, quando e da chi.

1.2.2 Il Service Value System

Il Service Value System (SVS) rappresenta il modo in cui i vari componenti e le attività di un'organizzazione lavorano insieme, come un sistema, per consentire la creazione di valore.

Gli input chiave del SVS sono le opportunità e la domanda mentre l'output è il valore fornito da prodotti e servizi.

- Opportunità: possibilità di aggiungere valore per gli stakeholders o di migliorare in qualche modo l'organizzazione.
- Domanda: necessità di prodotti o servizi espressa da consumatori interni o esterni all'organizzazione.

Le opportunità e la domanda sono due input costanti nel sistema, ma l'impresa non accetta automaticamente tutte le opportunità o soddisfa tutta la domanda.

Ciascun SVS di un'organizzazione si interfaccia con quelli di altre organizzazioni andando a formare un ecosistema.

L'architettura di ITIL SVS promuove la flessibilità e scoraggia il lavoro a vuoto, per questo motivo le attività della catena del valore del servizio e le pratiche ITIL non formano una struttura fissa e rigida ma, piuttosto, si tende a combinare più flussi di valore per soddisfare le esigenze dell'organizzazione in una varietà di scenari. Attraverso il suo modello operativo, ITIL 4 supporta molti approcci lavorativi innovativi come l'Agile, DevOps e Lean ma anche processi più tradizionali e il Project Management.

I componenti principali del SVS ITIL sono:

• ITIL guiding principles

• Governance

• ITIL Service Value Chain

• ITIL practises

• Continual improvement

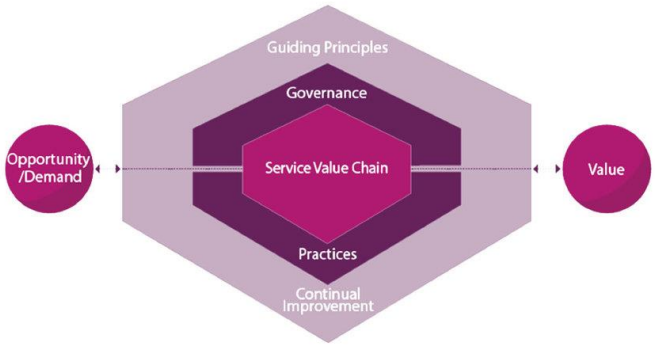


Figura 3 I componenti del SVS ITIL

Di seguito si esaminano in dettaglio i diversi elementi che compongono il Service Value System:

1) *ITIL guiding principles*

I principi guida di ITIL costituiscono un manuale di istruzioni e raccomandazioni universali che sono durature e aiutano l'organizzazione in ogni circostanza, indipendentemente dai cambiamenti nei suoi obiettivi, strategie e tipo di lavoro. Questi principi sono 7 ed hanno come principale scopo quello di incoraggiare i processi decisionali e promuovere il miglioramento continuo a tutti i livelli:

1. Focalizzarsi sul valore: ogni attività svolta deve portare, direttamente o indirettamente, valore per gli stakeholders;
2. Partire da dove ci troviamo: ITIL non richiede di partire con la creazione di qualcosa di nuovo ma di considerare quanto ci sia di disponibile e che possa essere sfruttato;
3. Avanzare iterativamente raccogliendo feedback: non bisogna fare tutto in un unico tentativo ma il lavoro deve essere organizzato in diverse sezioni ridotte e singolarmente gestibili;
4. Collaborare e promuovere la visibilità: le organizzazioni devono essere trasparenti e condividere il più possibile con gli altri, tenendo costantemente traccia del flusso di lavoro;
5. Pensare e lavorare in modo olistico: bisogna riconoscere che tutto è collegato. Nessuna pratica, dipartimento o fornitore è indipendente e tutte le attività devono essere allineate per arrivare all'obiettivo finale di creazione di valore;

6. Mantenere semplicità e praticità: si segue come principio “meglio fare meno cose, ma farle meglio”;
7. Ottimizzare e automatizzare: le organizzazioni devono massimizzare il valore del lavoro svolto aiutandosi, ove possibile, con l’automazione per effettuare task ripetitivi. L’automazione permette di ridurre il lavoro umano un-skilled per permettere a tali persone di lavorare su task più complessi che contribuiscono concretamente alla creazione di valore.

2) *Governance*

La governance fa riferimento ai mezzi con cui è diretta e controllata un’organizzazione. Inoltre, definisce le politiche e le regole comuni che l’organizzazione utilizza per fornire e mantenere i propri servizi.

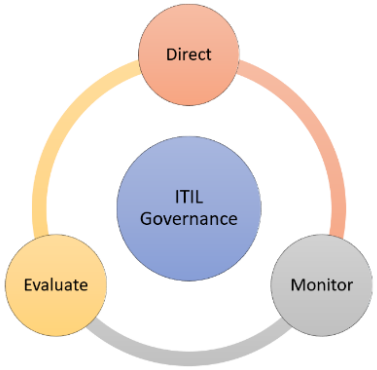


Figura 4 Le tre attività dell'ITIL Governance

Le tre attività principali che svolge la governance sono:

- Direct: assegnazione delle responsabilità all’interno dell’organizzazione ed attuazione di strategie e politiche organizzative.
- Monitor: monitoraggio delle performance e delle pratiche dell’organizzazione per definire se sono in linea con la strategia e le aspettative prefissate.
- Evaluate: esecuzione di revisioni regolari dell’organizzazione, della sua strategia, dei portafogli e delle relazioni con gli stakeholders (considerando anche cambiamenti dovuti a circostanze esterne).

3) *ITIL Service Value Chain*

L’elemento centrale di SVS è la sua Service Value Chain, ossia un modello operativo che delinea le attività chiave necessarie per creare valore, in risposta alla domanda, attraverso la creazione e la fornitura di prodotti e servizi.

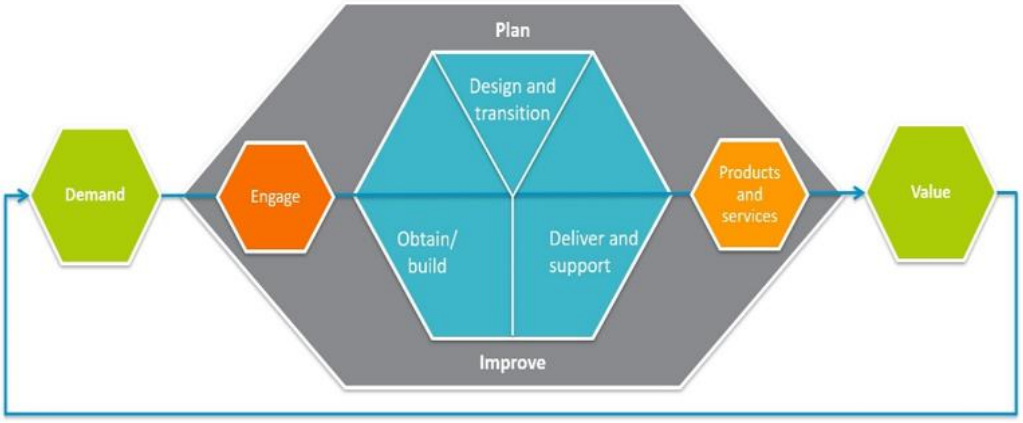


Figura 5 Le attività della Service Value Chain

In ITIL V4, le 6 attività della Service Value Chain sono:

1. Plan: avere visibilità del piano di lavoro facilita la comprensione della vision, dello stato attuale e delle pratiche di miglioramento da mettere in campo.

2. Engage: fornisce una buona comprensione delle esigenze degli stakeholders. Questa attività prende i requisiti dai clienti e li trasforma in requisiti di progettazione.

3. Design and transition: la progettazione assicura che i servizi e i prodotti soddisfino le aspettative degli stakeholders, considerando qualità, costi e time-to-market. L'obiettivo principale è quello di catturare i requisiti ricavati nella fase di Engage e fornire le specifiche per l'attività di Obtain/build.

4. Obtain/build: questa attività assicura che tutti i componenti del servizio siano disponibili (dove e quando necessario) e che essi soddisfino le specifiche concordate.

5. Deliver and support: assicura che i servizi siano erogati e supportati secondo le specifiche concordate e le aspettative degli stakeholder.

6. Improve: con lo scopo di migliorare continuamente prodotti, servizi e pratiche in tutte le attività della Service Value Chain.

4) *ITIL practises*

Mentre le versioni precedenti di ITIL si concentrano principalmente sui processi, ITIL 4 sposta il focus sulle pratiche che danno all'organizzazione maggiore flessibilità per:

- Implementare processi specifici che siano strettamente allineati alle esigenze dei clienti;
- Innovare nuovi processi per poter andare incontro a modalità moderne di lavoro con DevOps.

La pratiche in totale sono 34 e sono raggruppate in the categorie differenti:

1. Generali: per la gestione aziendale generale;
2. Di servizio: sviluppate per il service management e le industrie ITSM (IT service management);
3. Tecniche: spostano il focus su soluzioni tecnologiche.

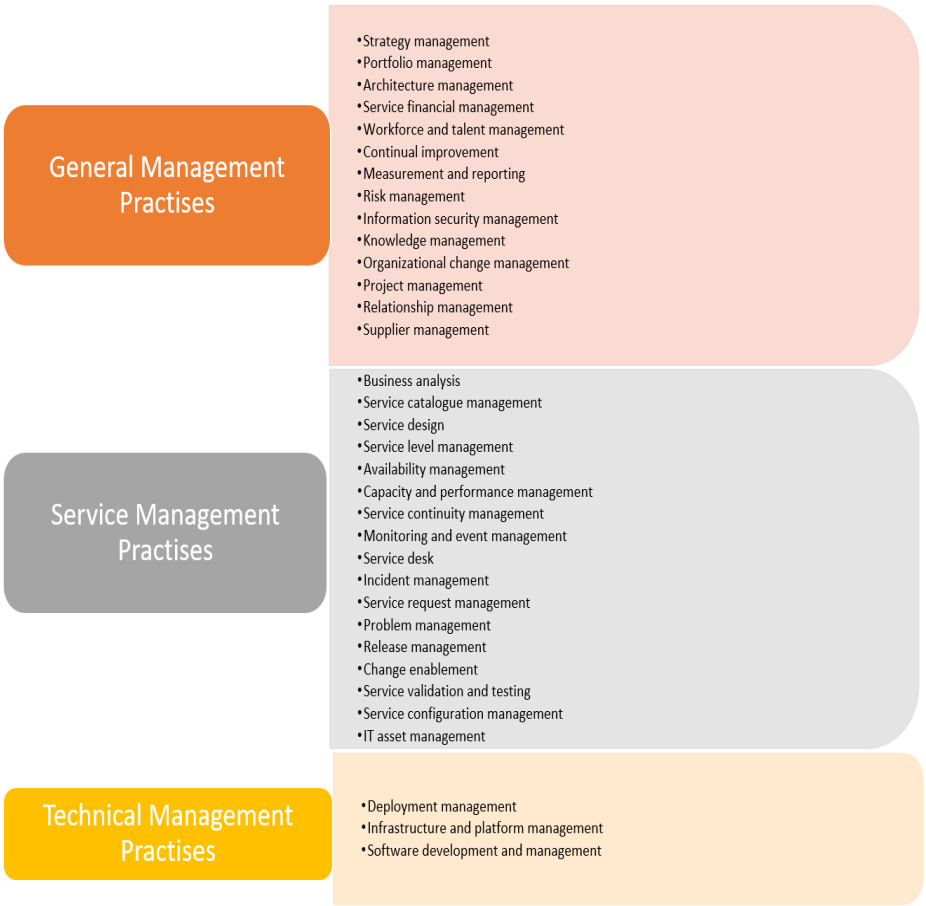


Figura 6 ITIL Practises

5) *Continual improvement*

Il modello di miglioramento continuo o CSI (Continual Service Improvement) è usato come guida di alto livello in tutte le aree di un'organizzazione e su tutti i livelli, dal livello strategico a quello operativo.

L'uso del modello aumenta la possibilità che le iniziative ITSM abbiano successo: esso si concentra sul valore del cliente e fa in modo che tutte le iniziative di miglioramento siano collegate alla vision dell'organizzazione.

Il modello è applicabile a iniziative molto piccole, come il miglioramento del servizio o delle operazioni, ma anche a cambiamenti organizzativi: l'approccio rimane analogo mentre le tecniche possono differire a seconda della dimensione dell'iniziativa.

Le attività chiave che fanno parte delle pratiche di miglioramento continuo includono i seguenti aspetti:

- Incoraggiare il miglioramento continuo all'interno dell'organizzazione;
- Garantire tempo e budget per il miglioramento;
- Identificare e registrare opportunità di miglioramento;
- Valutare e stabilire le priorità delle opportunità di miglioramento;
- Pianificare e attuare miglioramenti;
- Misurare e valutare gli eventuali risultati del miglioramento;
- Coordinare le attività di miglioramento all'interno dell'organizzazione.

Per tracciare e gestire le idee di miglioramento, dall'identificazione fino all'azione finale, le organizzazioni utilizzano un database o un documento strutturato chiamato registro di miglioramento continuo (continual improvement register). Poiché le aziende dipendono maggiormente dai servizi IT, è fondamentale che essi vengano continuamente migliorati, per soddisfare e raggiungere gli accordi sul livello di servizio.

Figura 7 Il ciclo di Deming

Nel 1980, lo statistico americano Deming, ha sviluppato un approccio di miglioramento definito il Plan-Do-Check-Act che si applica molto bene in ambito di CSI.

Il ciclo di Deming si articola in quattro fasi distinte che si ripetono, in maniera ciclica, fino al raggiungimento del miglioramento della qualità e della produttività.

Le 4 fasi sono:

1) PLAN (o fase della pianificazione): la prima fase del ciclo prevede l'esecuzione di un piano: monitoraggio dei sistemi, individuazione dei problemi e stesura di un piano di intervento per la risoluzione delle criticità.
2) DO (o fase dell'esecuzione): in questa fase si passa dal piano in forma teorica alla sua applicazione pratica mettendo in campo le pratiche individuate nella fase precedente. Allo stesso tempo di raccolgono i dati elaborati nel corso dell'esecuzione che saranno necessari per lo step successivo.
3) CHECK (o fase del controllo): durante la fase del controllo si analizzano accuratamente i dati statistici ricavati nel corso della messa in atto del piano di miglioramento ed, inoltre, si valutano tutte le eventuali correzioni da apportare al piano per integrarlo.
4) ACT (o fase dell'Azione): l'ultima fase prevede l'applicazione del piano, integrato con le eventuali modifiche. Al termine del ciclo si può decidere di applicarlo nuovamente, anche in caso di successo, in modo tale da mantenere costante nel tempo la qualità e il miglioramento.

1.3 Il ciclo di vita della gestione del servizio

Il nucleo principale ITIL consiste in 5 pubblicazioni, relative alle 5 fasi del ciclo di vita dei servizi IT:

1. La Service Strategy che riguarda la progettazione, lo sviluppo e l'implementazione della gestione dei servizi, sia come capacità organizzativa, che come risorsa strategica;

2. La Service Design che si concentra sulla garanzia che i servizi IT offerti soddisfino gli obiettivi dell'azienda e del cliente;

3. La Service Transition che è incentrata sulla gestione del rischio, della conoscenza, del cambiamento, delle risorse e della configurazione del servizio (e delle aree correlate).

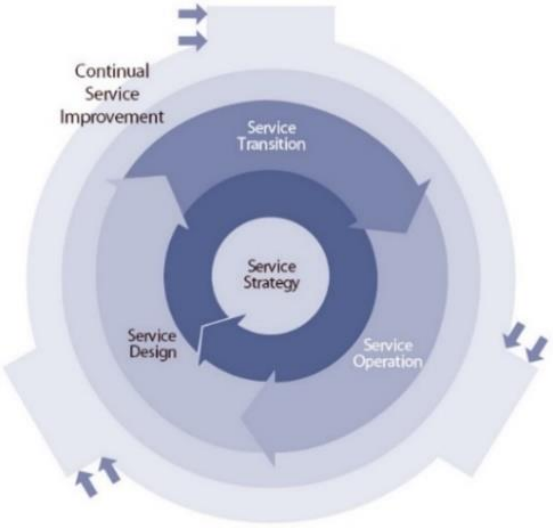


Figura 8 Il ciclo di vita della gestione del servizio

4. La Service Operation che si impegna a gestire quotidianamente la fase operativa dei servizi IT, garantendone sempre la disponibilità.

5. La fase del Continual Service Improvement che serve per migliorare l'esperienza dell'utente e la qualità dei servizi IT esistenti, ed è incorporata nelle quattro fasi descritte precedentemente.

1.4 Financial Management

Il processo di Financial Management ha come scopo quello di studiare e gestire i costi dell'infrastruttura IT e fornire una base finanziaria stabile per le decisioni aziendali relative all'area IT andando ad identificare e analizzare i costi di erogazione dei servizi.

Un processo di Financial Management si occupa quindi di:

- effettuare la contabilità di gestione dell'area IT;
- assistere nella riduzione dei costi di gestione nel lungo termine, senza rinunciare alla qualità degli standard;

- identificare i costi reali dei servizi IT;
- spiegare in che modo i servizi IT forniscono valore aggiunto all'azienda;
- valutare il ROI (Return On Investment), cioè l'indice di redditività del capitale investito;
- influenzare il comportamento degli utenti, ad esempio incentivando l'uso di risorse non critiche.

La gestione finanziaria è divisa in tre fasi:

1. Budgeting;
2. IT accounting;
3. Charging.

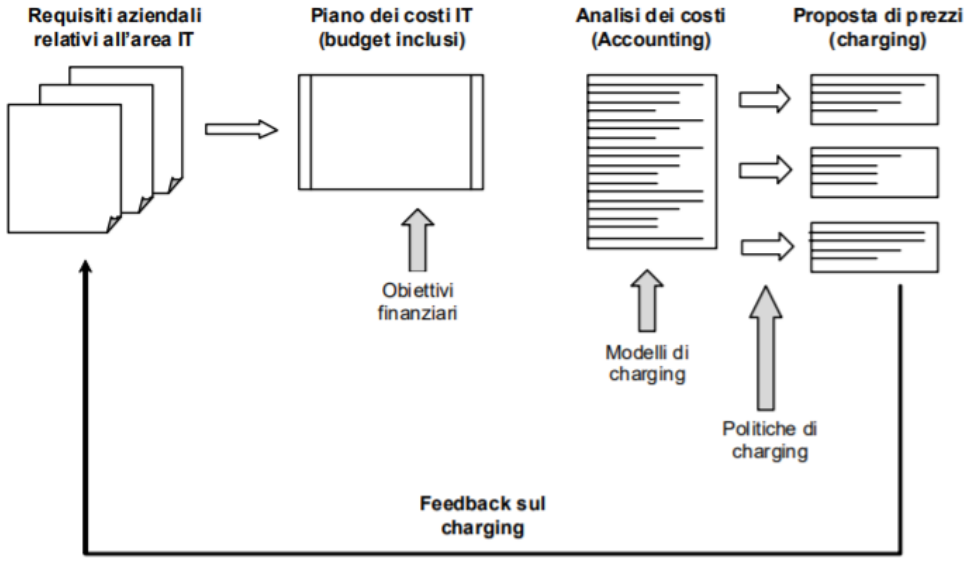


Figura 9 Fasi principali del Financial Management

Nella fase di *budgeting* si effettua il preventivo, il controllo e il monitoraggio dei costi in rapporto al bisogno espresso dal cliente, alle direzioni strategiche aziendali e al piano strategico dell'area IT. Questa fase solitamente consiste in un periodo di tempo (generalmente annuale) di previsione e di un successivo monitoraggio giornaliero delle spese in rapporto con le aspettative preventivate.

Nella fase di *IT accounting* viene effettuata una descrizione dettagliata sul denaro speso per la gestione dei servizi IT motivando, in termini amministrativi e di controllo di gestione, la natura della spesa IT.

Infine, nella fase di *charging* si permette all'organizzazione di recuperare i costi della fornitura dei servizi; ciò può avvenire con diversi modelli, ad esempio:

- cost recovery: in cui il prezzo del servizio è uguale al costo;
- cost recovery + markup: in cui il prezzo è uguale al costo più un ricarico (markup);
- market rate: in cui il prezzo è quello di mercato, quindi non collegato al costo reale sostenuto.

Si ricorda che i costi possono ovviamente essere di diversa categoria. In particolare possono essere:

- capitali o operativi;
- diretti o indiretti;
- fissi o variabili.

2. IL MODELLO DEVOPS

2.1 Introduzione al modello

DevOps è un movimento culturale e professionale che sottolinea la comunicazione, la collaborazione e l'integrazione tra gli sviluppatori di software e i professionisti dell'esercizio IT: come conseguenza il flusso di lavoro è migliore e permette alle organizzazioni la flessibilità necessaria per cambiare rapidamente, senza sacrificare la qualità e l'affidabilità dei servizi basati sull'IT.

Il termine DevOps (Development and Operations) è stato reso popolare in Belgio a partire dal 2009; in generale, "Dev" rappresenta tutte le persone coinvolte nello sviluppo di prodotti e servizi software (inclusi rappresentanti aziendali e fornitori) mentre "Ops" comprende tutte le persone coinvolte nell'erogazione e gestione di tali prodotti e servizi.

Lo State of Devops Report, pubblicato nel 2015, ha rilevato che le aziende che stanno facendo DevOps sono più performanti rispetto a quelle che non lo usano infatti, se i team e i diversi individui lavorano insieme, è favorevole per il business rispetto ad avere "silos" di ingegneri che non vogliono condividere le proprie conoscenze reciprocamente.

Nei silos si genera un accumulo di informazioni come forma di sicurezza: se un soggetto è l'unico a sapere fare X, egli diventa indispensabile e l'organizzazione non può fare a meno di lui; ciò però comporta difficoltà e lentezza nel completare un lavoro che coinvolge più squadre e diminuisce il morale perché i vari team iniziano a vedersi l'un l'atro come rivali.

Altri concetti che la cultura DevOps vuole superare sono:

- Blame culture: si tende a non incolpare o punire le persone quando commettono errori. Se una organizzazione punta sempre a trovare una persona o un gruppo da incolpare per ogni incidente che si verifica, i singoli individui saranno motivati a provare a spostare la colpa lontano da loro stessi. Questo tipo di ambiente non si presta bene ad una cultura aperta e collaborativa.
- Root Cause Analysis (RCA): è un metodo per identificare le cause radice di un problema per poter mettere in campo le azioni appropriate in ottica di

prevenzione. La RCA è un processo iterativo che però tende ad andare ad identificare solo le cause dirette piuttosto che un insieme di elementi che, insieme, possono contribuire.

- Errore umano: si è soliti a presumere che vengano commessi errori umani per semplice negligenza, affaticamento o incompetenza trascurando molteplici ulteriori fattori che contribuiscono alla persona nel momento in cui prende una decisione o compie una azione.

Le organizzazione più performanti che stanno usando DevOps distribuiscono il codice più velocemente, hanno meno guasti (e, in caso di guasto, recuperano più velocemente) ed hanno dipendenti più felici.

DevOps riconosce, inoltre, che l'evoluzione verso il cloud computing ed i dispositivi mobili intelligenti hanno portato a un cambiamento di paradigma in termini di modalità di sviluppo ed erogazione dei servizi IT. A loro volta, questi progressi hanno spostato le aspettative di business e dei clienti, sia in termini di rapidità nello sviluppo e nell'erogazione dei servizi IT, sia nell'affidabilità di tali servizi. Rimangono tuttavia alcuni ostacoli nell'adozione e nella diffusione delle pratiche DevOps, soprattutto a causa di budget limitati e timori in materia di sicurezza e cultura aziendale.

Nonostante il 76% delle organizzazioni italiane che applicano le pratiche DevOps abbia registrato un aumento del 32% della produttività dei dipendenti, incrementato del 36% l'efficienza gestionale, ridotto del 27% i costi associati all'IT e aumentato del 38% la customer satisfaction; da un sondaggio è stato rilevato che, mentre l'84% dei partecipanti ritiene fondamentale acquisire le competenze informatiche necessarie per mettere in pratica DevOps, solo il 35% le ha già adottate (dati tratti dall'indagine voluta da CA Technologies e condotta da Freeform Dynamics).

L'evoluzione dei servizi digitali sta mostrando scenari competitivi impensabili fino a pochi anni fa, per le organizzazioni della "new economy" è decisamente più semplice iniziare fin da subito con un approccio DevOps perché è il modello che consente loro di arrivare prima sul mercato. Al contrario, le organizzazioni tradizionali oggi devono indubbiamente fare i conti con la complessità del cambiamento: è comune affermare che adottare ITIL è "ancora il modo migliore per gestire una modifica, perché tutti sanno

cosa fare e non si confondono con i ruoli", ma è necessario *adattare* il framework ITIL per poter collaborare internamente secondo l'approccio DevOps.

Il movimento DevOps è uno sforzo per trasformare il modo in cui le varie parti interessate interagiscono, comunicano e lavorano insieme nel ciclo di vita dello sviluppo software.

John Willis, autore di sei libri sulla gestione dei sistemi aziendali, sottolinea che il motivo per cui le persone ignorano DevOps come qualcosa di nuovo o degno di attenzione è la mancanza di una "definizione canonica". A differenza di Agile, infatti, non esiste un "manifesto DevOps" che associ il movimento a un luogo di nascita e un tempo specifico o che fornisca un insieme specifico di valori, metodi, pratiche e strumenti: DevOps è divenuto popolare nel 2009 grazie ad una presentazione di John Allspaw e Paul Hammond alla conferenza Velocity, chiamata: 10+Deploy Per Day: Dev and Ops Cooperation presso Flickr.

Alcune delle frasi riportate nella presentazione originale sono le seguenti:

- "In the traditional thinking Dev's job Is to add new features and Ops'job is to keep the site stable and fast."
- "Ops' job is NOT to keep the site stable and fast, Ops' job is to enable the business (this is Dev's job too). The business requires change, but change is the root cause of most outages! Discourage change in the interests of stability or allow change to happen and often as it needs to.
  Lowering risk of change through tools and culture".

Debois riteneva che DevOps fosse una risposta all'inflessibilità e alla mentalità "noi contro loro" che le organizzazioni hanno creato mettendo "contro" sviluppatori di software, tester, manager, amministratori di database, tecnici di rete, amministratori di sistema, ecc.

La tecnologia ha poi contribuito a creare una convergenza armonica dei principi di gestione DevOps e degli strumenti software in grado di introdurre procedure DevOps critiche, come la collaborazione, l'automazione, il monitoraggio, la registrazione e la distribuzione di software. Utilizzando una combinazione di strumenti software e una

cultura di collaborazione e miglioramento costante, è nato il movimento DevOps per lo sviluppo di software.

Negli anni si sono susseguite diverse definizioni di DevOps, una comune e molto sintetica afferma che: "per DevOps si intende un insieme di pratiche intese a ridurre il tempo che intercorre tra il commit di una modifica in un sistema e l'avvenuta modifica in produzione, garantendo al contempo un'alta qualità. (Len Bas, Ingo Weber,Liming Zhu (2015). Chapter 1 "What is DevOps?""DevOps A Software Architect's Perspective)

2.2 Le tre vie di DevOps

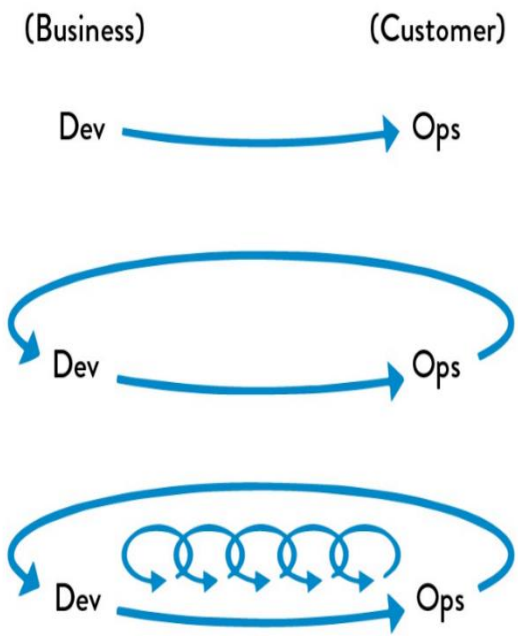


Figura 10 Le tre vie di DevOps

1. System thinking: la prima via enfatizza le dimensioni dell'intero sistema senza interessarsi alle prestazioni degli specifici reparti. L'attenzione si concentra su tutti i flussi di valore aziendali abilitati dall'IT.

2. Amplify feddback loops: la seconda via riguarda la creazione di circuiti di feedback da destra a sinistra con l'obiettivo di poter apportare continuamente le correzioni necessarie comprendendo le richieste di tutti i clienti, interni ed esterni.

3. Culture of continual experimentation and learning: la terza via riguarda la creazione di una cultura che promuova la sperimentazione continua, l'assunzione di rischi e l'apprendimento dal fallimento.

2.3 L'etica DevOps: Developer and Operations

Come già affermato precedentemente, prima di DevOps, il team di sviluppo e il team operazionale lavoravano in modo completamente distaccato. Test e implementazione erano attività isolate che venivano eseguite solo dopo la progettazione e, quindi, consumavano più tempo rispetto ai cicli attuali di build.

Senza utilizzare DevOps:

• I team di sviluppo e il team operazionale avevano tempistiche distinte e non sincronizzate che causavano ulteriori ritardi. Nello specifico, Developer e Operations hanno i loro processi, strumenti e basi di conoscenza indipendenti.

L'interfaccia tra loro, nell'era pre-DevOps, era di solito un sistema di ticket: i team di sviluppo richiedevano l'implementazione di nuove versioni del software e il personale operativo gestiva manualmente quei ticket. In questo accordo però, i team di sviluppo cercavano continuamente di spingere le nuove versioni in produzione, mentre il personale operativo tentava di bloccare queste modifiche per mantenere la stabilità del software.

Teoricamente, questo modus operandi offre una maggiore stabilità per il software in produzione però, nella pratica, comportava lunghi ritardi tra gli aggiornamenti del codice e la distribuzione (oltre a processi di risoluzione dei problemi inefficaci):

• i membri del team impiegavano molto tempo a testare, distribuire e progettare invece di costruire il progetto;

• nessuna pratica di programmazione estrema comprendeva la distribuzione e, di conseguenza, il passaggio alla produzione tendeva ad essere un processo stressante nelle organizzazioni, infatti venivano svolte attività manuali soggette a errori e, anche, correzioni dell'ultimo minuto.

Molto spesso i vari team operano in una atmosfera tesa, in cui la mancanza di dialogo e collaborazione impattano negativamente sulla soddisfazione del cliente e sul business in generale. Accade, ad esempio, che chi si occupa dei test o della sicurezza viene coinvolto solamente in un secondo momento, quando è già troppo tardi per risolvere

problemi ed errori facilmente evitabili. Il risultato consiste in tempi di stallo e riavvio del processo, in frustrazione e sconforto da parte di chi cerca invano risultati soddisfacenti.

Tra il 1970 e il 1980, si impiegavano anni prima di rilasciare un prodotto software. A partire dagli 2000, grazie alle nuove tecnologie e all'adozione dei principi Agile, le tempistiche di sviluppo e deployment si sono ridotte a mesi o addirittura settimane.

Infine, a partire dal 2010, con l'avvento del DevOps, è stato possibile utilizzare il software in produzione in poche ore o addirittura minuti, massimizzando l'efficienza e minimizzando il rischio di ottenere un funzionamento indesiderato. Le compagnie che non adottano il DevOps, invece, non sono in grado di effettuare centinaia o migliaia di deployment giornalieri ma necessitano di mesi o quadrimestri; questo è dovuto principalmente alla loro incapacità di risolvere il conflitto cronico Development vs Operations all'interno della loro organizzazione e comporta un forte rialzo del cosiddetto "technical debt". Quest'ultimo è una metafora inventata da Ward Cunningham per descrivere come decisioni sbagliate comportino problemi sempre più difficili da rimediare nel tempo, diminuendo il numero di soluzioni disponibili. Il fattore critico che contribuisce ad aumentare il technical debt è sicuramente quello degli obiettivi discordanti dei team di sviluppo e di quelli che si occupano della configurazione ed effettiva messa in produzione del software: mentre i primi hanno come priorità quella di rispondere rapidamente al cambiamento dei requisiti in risposta alle esigenze di mercato, i secondi, d'altro canto, hanno come priorità quella di fornire un servizio stabile, affidabile e sicuro al cliente (a volte diminuendo drasticamente la flessibilità della soluzione).

Come risultato, il ciclo di delivery del software avanza sempre più lentamente e sempre meno progetti vengono intrapresi, perdendo terreno nei confronti del mercato che si evolve costantemente.

2.4 Il ciclo di vita di DevOps (plan – code – build- test- release- deploy – operate – monitor)

Le fasi del ciclo di vita di DevOps sono le seguenti:

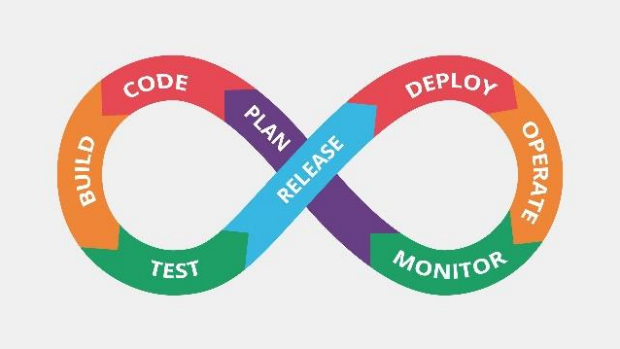


Figura 11 Il ciclo di vita di DevOps

1. Continuos Development
2. Continous Integration
3. Continous Testing
4. Continuos Monitoring
5. Continuos Feedback
6. Continuos Deployment
7. Continuos Operations

1. CONTINOUS DEVELOPMENT

Questa fase comporta la pianificazione e la fase di coding del software. La vision del progetto è decisa durante la pianificazione e gli sviluppatori possono iniziare a sviluppare il codice per l'applicazione.

Il codice può essere scritto in un qualsiasi linguaggio.

2. CONTINOUS INTEGRATION

In questa fase gli sviluppatori eseguono modifiche al codice sorgente, eseguono gli unit test (per assicurarsi che ogni funzionalità sia funzionale se presa singolarmente) e gli integration test (per verificare se le varie funzionalità si integrano tra di loro o se vanno in conflitto).

Il codice che supporta nuove funzionalità è continuamente integrato con il codice già esistente.

3. CONTINOUS TESTING

Al fine di evitare la presenza di bug, il codice viene testato in maniera costante attraverso degli strumenti appositi come TestNG oppure Selenium; essi permettono agli sviluppatori di testare più codici contemporaneamente in parallelo con il compito di assicurarsi che non ci siano difetti nelle funzionalità.

Tali test automatici permettono di risparmiare molto tempo e fatica rispetto ai test manuali inoltre consentono di identificare più facilmente i failures e consentono di programmare l'esecuzione dei test cases in momenti predefiniti.

4. CONTINOUS MONITORING

Fase di monitoraggio continuo delle prestazioni dell'applicazione in cui, solitamente, vengono identificate le cause dei principali problemi (anche gli eventuali problemi di rete vengono risolti in questa fase).

Questa pratica necessita l'intervento di un team operativo che monitori gli errori commessi dagli utenti e qualsiasi comportamento improprio del sistema; gli strumenti più popolari che accompagnano l'intervento umano in modo proattivo sono Splunk ed ELK Stack.

L'obiettivo è quello di monitorare le prestazioni dell'applicazione secondo i requisiti del cliente ed apportare eventuali modifiche per soddisfarli.

5. CONTINOUS FEEDBACK

Lo sviluppo dell'applicazione viene costantemente migliorato analizzando i risultati prodotti dal software: la continuità è il fattore essenziale che permette di rimuovere alcuni step tradizionali che vanno da una versione del software a quella successiva.

6. CONTINOUS DEPLOYMENT

In questa fase il codice viene distribuito ai server di produzione. È importante mantenere la coerenza tra gli ambienti in cui viene testata l'applicazione e gli ambienti in cui viene sviluppata e distribuita: in questo modo, nell'ambiente di produzione, non vi è possibilità di errore poiché il prodotto è già stato sperimentato in modo analogo in altri ambienti che non impattano direttamente l'operato del cliente.

Un tipico processo di deployment è il seguente:

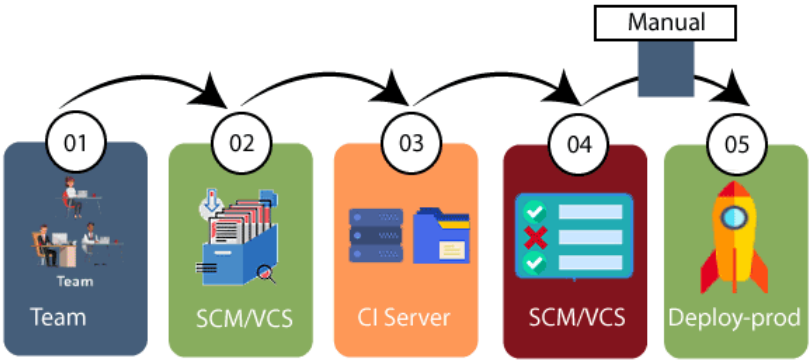


Figura 12 Processo di deployment DevOps

dove:

SCM = Source Control Management

CI Server = Continuous Integration Server

7. CONTINOUS OPERATIONS

Tutte le operations DevOps si basano sulla continuità e sulla completa automazione del processo di rilascio che permettono all'organizzazione di accelerare il time to market complessivo.

2.5 I microservizi

L'implementazione efficace degli strumenti e delle procedure DevOps spesso richiede una architettura software nota con il nome di microservizi.

I microservizi sono un approccio per sviluppare e organizzare l'architettura dei software secondo cui, questi ultimi, sono composti di servizi indipendenti di piccole dimensioni che comunicano tra di loro tramite API ben definite.

Questi servizi sono controllati da piccoli team autonomi e tale struttura permette di scalare e sviluppare le applicazioni in modo più semplice e rapido, accelerando il time-to-market di nuove funzionalità.

Tramite i microservizi, le applicazioni vengono scomposte nei loro elementi più piccoli (indipendenti l'uno dall'altro): ciascun componente, o processo, rappresenta un microservizio.

2.5.1 Architettura monolitica vs architettura di microservizi

Nelle architetture monolitiche tutti i processi sono strettamente collegati tra loro e vengono eseguiti in modo sequenziale. Ciò significa che, qualora si volesse limitare un processo dell'applicazione che esegue un numero elevato di richieste, sarebbe necessario ridimensionarla interamente: aggiungere, eliminare o migliorare una singola funzione dell'applicazione monolitica risulta più complesso in quanto richiede un riadattamento di tutto il progetto.

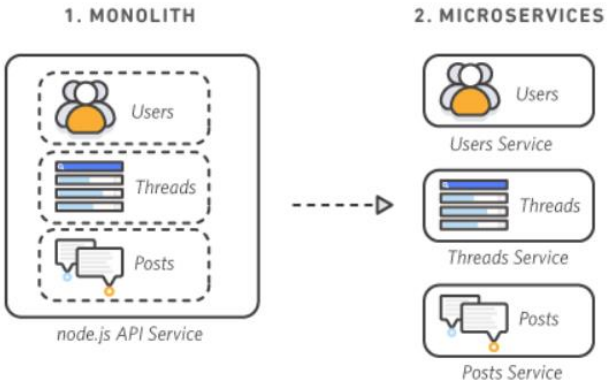


Figura 13 Architettura monolitica vs architettura di microservizi

Al contrario, ciascun servizio nell'architettura basata su microservizi può essere sviluppato, distribuito, eseguito e ridimensionato senza influenzare il funzionamento degli altri componenti e ciascun servizio è progettato per una serie di capacità e si concentra sulla risoluzione di un problema specifico (se, nel tempo, gli sviluppatori

aggiungono del codice aggiuntivo a un servizio rendendolo più complesso, il servizio può essere scomposto in servizi più piccoli).

In linea generale i microservizi offrono:

1. Eliminazione di “single point of failure”: un’applicazione scomposta in microservizi riduce notevolmente il rischio che un errore di codice o configurazione si riversi sull’intero sistema.
2. Amministrazione più elastica: lo sviluppo dei processi (build, test, deploy, update) può essere gestito in modo agile e semplice poiché gli ambienti di sviluppo, test e produzione rimangono più coerenti e allineati tra loro.
3. Scalabilità efficace: i microservizi consentono di scalare ciascun servizio in modo indipendente per rispondere alla richiesta delle funzionalità che l’applicazione supporta.
4. Codice riutilizzabile: Dividere il software in moduli piccoli e ben definiti permette ai team di utilizzare funzioni per più scopi; un servizio scritto per una certa funzione può essere utilizzato come blocco costruttivo per un’altra funzionalità.
5. Libertà del linguaggio di sviluppo e degli strumenti di lavoro: il pattern dei microservizi elimina ogni vincolo preso a lungo termine sulla scelta del linguaggio.

   Quando si implementa un nuovo servizio, gli sviluppatori sono liberi di scegliere il linguaggio di programmazione e framework corrispondente più idonei al nuovo servizio e, di conseguenza, i team possono scegliere il miglior strumento per ciascun lavoro.

2.6 L’efficacia della gestione dei Change con DevOps

I dati pubblicati nello State Of DevOps Report 2020, rivelano l’impatto significativo del modello DevOps sugli aspetti organizzativi e sulle performance di una infrastruttura IT.

Nel 2020 il sondaggio ha interessato oltre 2.400 partecipanti in tutto il mondo che lavorano nei settori IT, sviluppo, sicurezza delle informazioni e aree correlate.

I punti chiave introduttivi che emergono sono i seguenti:

- La creazione di valore attraverso un software non dipende esclusivamente da una buona collaborazione tra sviluppatori e operatori ma quasi tutte le funzioni aziendali adiacenti fanno parte del processo e devono evolversi;
- DevOps ha, fondamentalmente, come obiettivo quello di consentire alle persone di collaborare tra loro verso un obiettivo aziendale comune.
  Per poter fare ciò è quindi necessaria una buona comunicazione ed un flusso libero di lavoro ma, ancora oggi, uno dei motivi principali per cui DevOps spesso non riesce ad espandersi è che la maggior parte delle aziende non fornisce i giusti incentivi: ciò genera mancanza di responsabilità e proprietà nei confronti dei risultati aziendali;

Nell'ultimo decennio, le pratiche DevOps hanno stravolto il modo in cui lavorano i team che operano in ambiti IT. La tabella seguente illustra i cambiamenti più importanti:

Prima di DevOps	Dopo DevOps
Progetti Waterfall e release molto pesanti	Piccoli lotti sono deliverati al clienti frequentemente portando ad implementazioni più frequenti e cycle time più veloci
Cicli di approvazioni lenti: numerose revisioni ed approvazioni che portano a lunghi tempi di attesa	Feedback e KPI in tempo reale guidati da flussi di lavoro automatizzati
Gestione complessa e dispendiosa nel caso in cui si debbano applicare dei Change	Coinvolgimento tempestivo di tutte le parti necessarie per mettere in atto i Change
I team sono organizzati secondo confini funzionali: gli incentivi non sono allineati	I team sono allineati agli obiettivi aziendali

Per vedere se l'efficacia della gestione dei Change è correlata all'evoluzione di DevOps sono prese in considerazione 3 dimensioni:

1. Successo di implementazione: si esamina il tasso di errore dei Change e la frequenza di deployment. Idealmente, le aziende dovrebbero essere in grado di apportare modifiche frequentemente, riprendersi rapidamente dagli errori ed imparare da essi.

2. Livello di efficienza: si valuta l'efficienza di un processo di gestione dei Change sulla base di quanto segue:
    - Le modifiche richiedono una sola approvazione;
    - Le modifiche sono implementate correttamente e non devono essere annullate;
    - Il tempo necessario per documentare il Change è breve.
3. "Sentimento" delle prestazioni: è stato chiesto agli intervistati se le procedure adottare avessero:
    - Ridotto i rischi;
    - Ridotto i tempi di inattività relativi a malfunzionamenti del sistema (Incident);
    - Fornito informazioni utili all'organizzazione;
    - Fornito un livello di revisione e approvazione appropriato (in base al livello di rischio).

È emerso che l'efficacia delle gestione dei Change aumenta man mano che le organizzazioni evolvono le loro pratiche DevOps e, sebbene le differenze non siano enormi, risultano statisticamente significative.

I risultati sono riassunti nel seguente grafico:

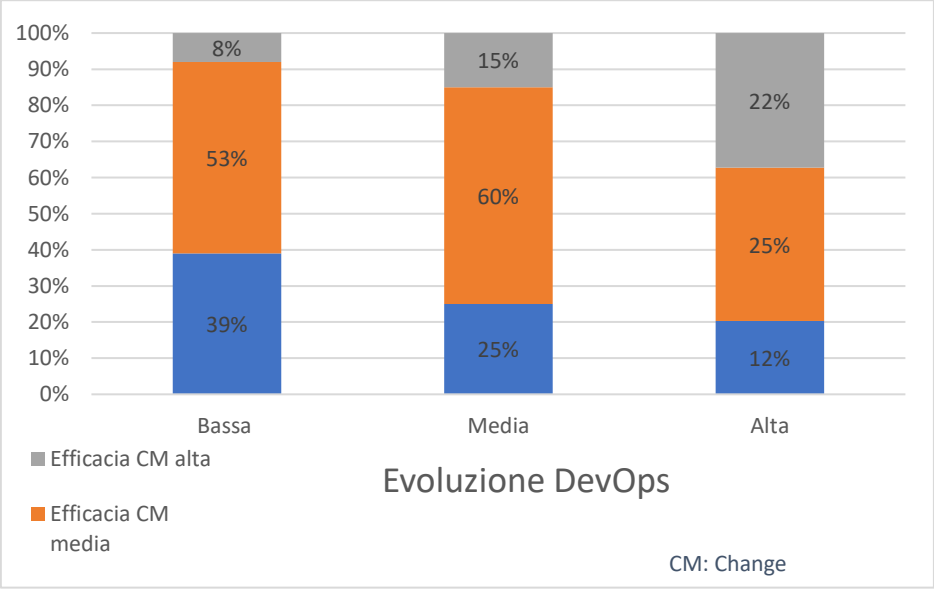


Figura 14 Evoluzione DevOps vs Efficacia CM

Un ulteriore risultato interessante che emerge dalla analisi è il seguente: all'aumentare del livello di automazione durante la fase di test e deployment, aumenta la percentuale del "sentimento" delle prestazioni.

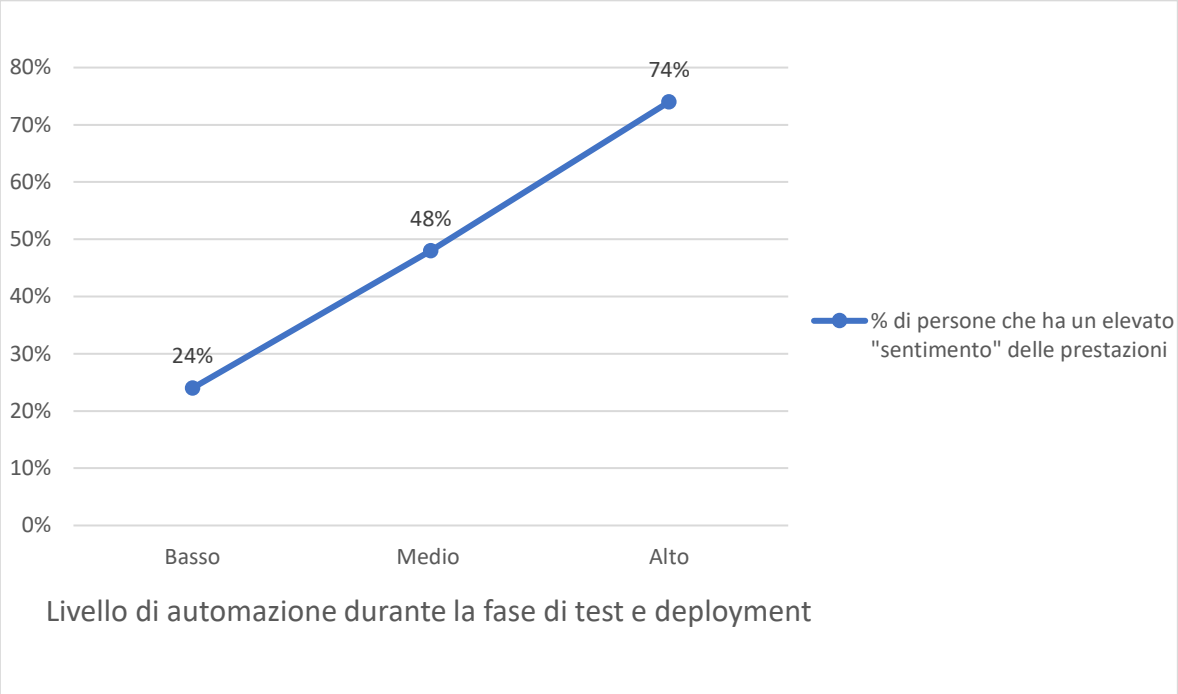


Figura 15 Livello di automazione vs % di persone con "sentimento" delle prestazioni elevato

Questo implica che l'automazione dà fiducia ai team durante la gestione dei Change principalmente perchè:

- Riduce il rischio e fornisce un livello appropriato di revisione;
- Riduce i tempi di inattività dei servizi;
- Fornisce informazioni utili per tutta l'organizzazione;
- Assicura che le conoscenze e le informazioni siamo condivise con tutte le parti interessate.

I partecipanti al sondaggio provengono da una vasta gamma di nazioni, settori e dimensioni aziendali.

Alcune delle informazioni più significative sono riportate nei seguenti grafici:

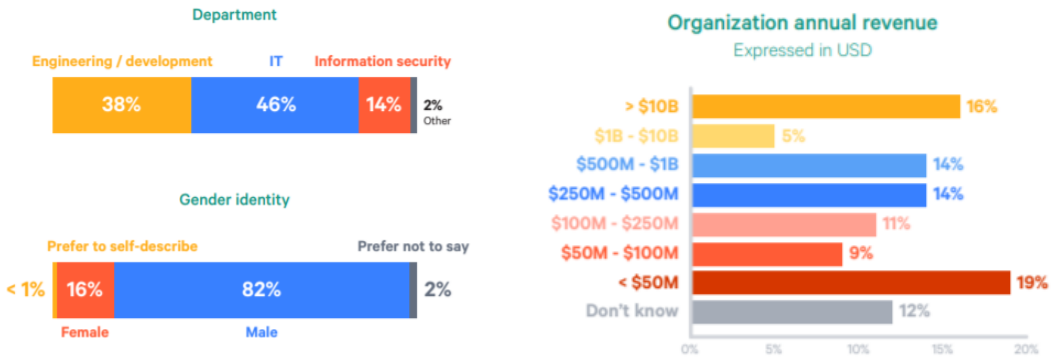


Figura 16 Informazioni sul campione analizzato per il sondaggio

2.7 Confronto tra DevOps, Agile e ITIL

È comune considerare Agile e DevOps come due metodologie distinte quando, in realtà, DevOps non è altro che un miglioramento recente della metodologia Agile.

DevOps estende i principi della metodologia Agile che si fondano su valori, metodi e pratiche, concentrandosi sull'intero servizio, ed aggiungendo gli strumenti. Secondo il Manifesto Agile, infatti, gli strumenti assumono una minima importanza, ma ciò ha portato nel tempo a trascurarli notevolmente con conseguenze spesso non in linea con l'obiettivo del cliente.

Un'altra differenza riscontrata tra i modelli è che, mentre Agile risolve i problemi della tecnologia, il modello DevOps cerca di risolvere un problema di business attraverso un insieme di pratiche, cultura e valori che portano ad avere un software più solido e che viene prodotto più velocemente, con meno problemi durante il ciclo di vita.

Per fare in modo che ciò si realizzi, infatti, DevOps accumuna l'obiettivo di Developers e Operators, puntando all'ottimizzazione dell'intero sistema invece che alle ottimizzazioni locali; ciò implica che al raggiungimento degli obiettivi di business, sarà risolto il problema del cliente, sia esso un problema di Developer o di Operation.

Un ulteriore confronto possibile è quello tra DevOps ed il framework ITIL (analizzato nel Capitolo 1): entrambe sono filosofie che hanno come scopo quello di incrementare il valore ed, infatti, è lo stesso DevOps Handbook a dichiarare che "ITIL e DevOps possono essere compatibili. Per supportare le alte frequenze di deployment associate con

DevOps, però, molte aree dei processi di ITIL dovrebbero diventare completamente automatizzate, risolvendo così molti problemi associati alla gestione delle configurazioni e dei rilasci."

Quindi, seppur lo scopo sia comune, l'incremento di valore desiderato si ottiene in modo differente:

- In ottica DevOps si ottiene tramite il miglioramento della collaborazione e del coordinamento all'interno di un'organizzazione, creando una migliore relazione di lavoro tra gli sviluppatori e i team operativi.

  Il punto di forza perciò è proprio la potenza della comunicazione e della cooperazione;
- In ottica ITIL, invece, il valore si ottiene standardizzando la struttura ITSM all'interno dell'organizzazione ed utilizzando in modo efficace le pratiche imposte.

In seguito una tabella riassuntiva che mostra graficamente l'impatto dei principali fattori critici di successo nelle diverse metodologie:

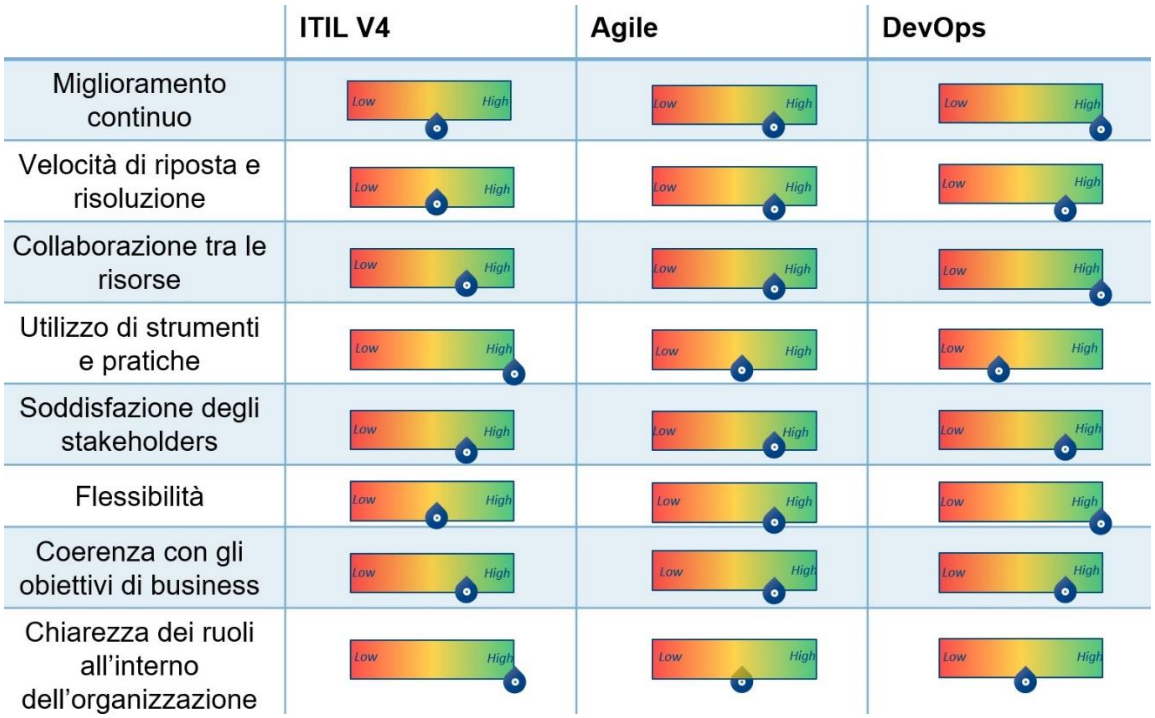


Figura 17 Confronto KSF nei diversi modelli

3. PRESENTAZIONE DELL'AZIENDA E ORGANIZZAZIONE AZIENDALE

3.1 Presentazione dell'azienda

Figura 18 Logo Techedge S.p.A.

Techedge S.p.A. nasce nel 2004 a Milano, grazie ad un'idea di Domenico Restuccia, fondatore e attuale amministratore delegato. Opera nel settore dei servizi, offrendo alle aziende la progettazione e lo sviluppo di software innovativi, permettendo di incrementare il loro valore ed individuare nuove opportunità di mercato. Sfruttando le migliori tecnologie aiuta i suoi clienti ad introdurre processi innovativi in modo rapido, riuscendo a coniugare la consulenza di processo e la competenza tecnologica. L'anno successivo possiede già cinque uffici distribuiti in Italia, Germania e Spagna.

Tra il 2008 e il 2012 la società si espande ulteriormente ampliando le proprie competenze e giungendo in Brasile e USA. Tra il 2012 e il 2016 arriva in Messico, Cile, Colombia, Perù, Arabia Saudita e Inghilterra, contando più di 1500 dipendenti, 10 startups e 150 clienti. Il 19 dicembre 2018 fa il suo ingresso in Borsa Italiana e nel 2019 ha realizzato ricavi per 98,7 milioni di euro. In Italia, ad oggi, conta più di dieci sedi operative, tra cui Milano (sede principale), Roma, Torino (sede in cui è stato svolto questo progetto di tesi), Catania, Asti, Pescara, Lucca, Cagliari, Padova e Bergamo.

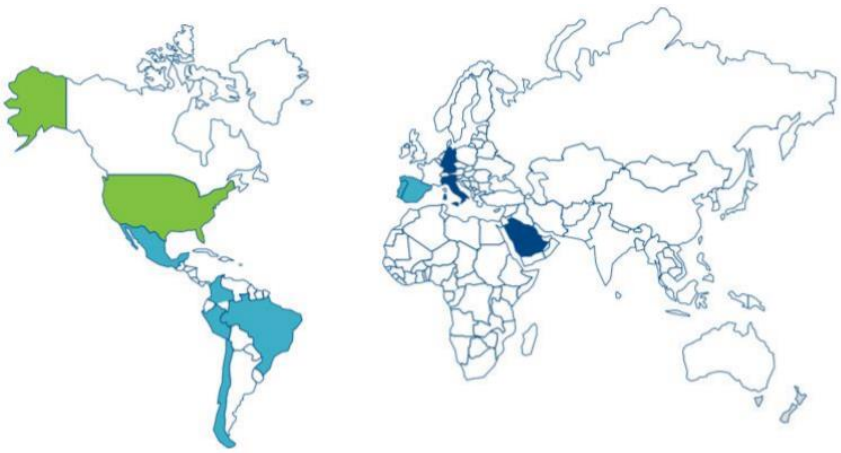


Figura 19 Sedi operative Techedge S.p.A

I settori in cui opera Techedge sono:

- Intelligence Operations: comprende tutte quelle attività e servizi tecnologici che generano valore aggiunto.
- Intelligence Enterprise: comprende il settore finanziario, data intelligence, gestione della supply chain e Industry 4.0.
- Intelligence Technology: comprende l'utilizzo delle piattaforme informatiche di SAP, Microsoft Oracle, AWS e Google Cloud.
- Industry: comprende la gestione della Customer operations e i settori industriali (servizi finanziari, manufacturing, energia, automotive, turismo, food & beverage, trasporti, gaming, aerospace, fashion & luxury ed healt care) .

Le aree specifiche su cui si posiziona l'ambito di un progetto svolto in Techedge sono dette Practice; se i progetti risultano molto complessi, può capitare che questi comprendano più di un'area specialistica e quindi vengono definiti Multi-Practice.

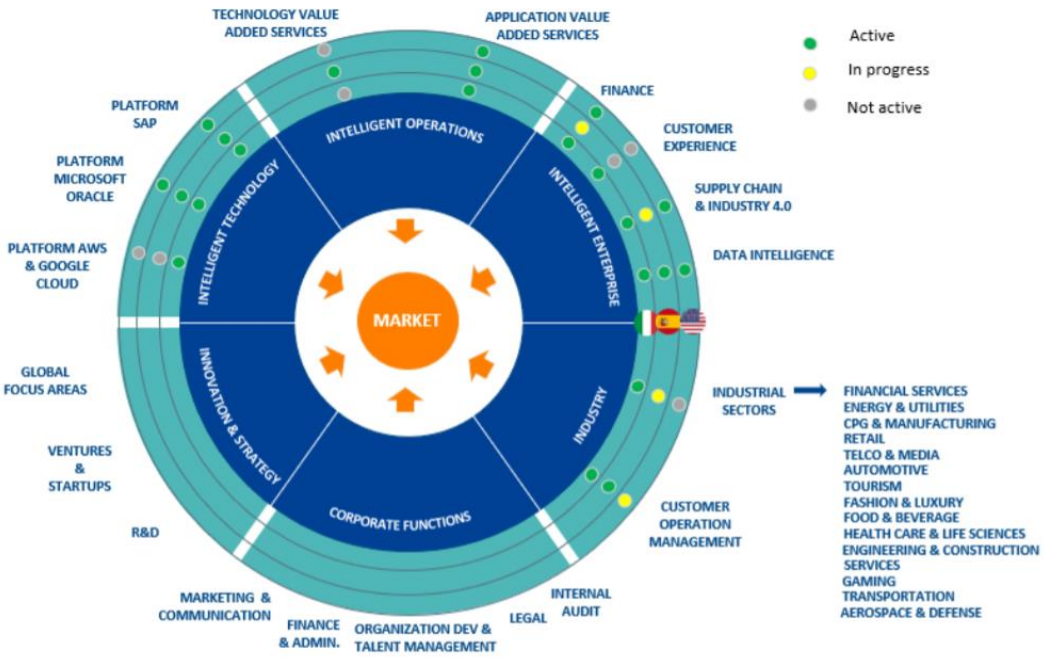


Figura 20 Rappresentazione dei settori in cui opera Techedge S.p.A.

3.2 Organizzazione aziendale

L'organizzazione di Techedge e le relazioni tra i dipendenti sono tra i punti di forza per poter garantire flessibilità, adattabilità alle esigenze del cliente e poter sviluppare contenuti altamente tecnologici ed innovativi. Ogni dipendente acquisisce costantemente nuove competenze fornendo soluzioni ai problemi legati allo sviluppo di software personalizzati e supportando la ricerca e sviluppo.

L'azienda risulta essere così strutturata:

1. Practice Manager: ha il ruolo di sviluppare competenze, strumenti e metodologie per assicurare l'ottimo svolgimento del progetto;
2. Deputy Practice Manager: figura di supporto al Practice Manager nella gestione delle attività quotidiane;
3. Project Manager: responsabile del corretto svolgimento dei progetti;
4. Consultant & Professional: costituiscono il team di esperti coinvolti nello svolgimento dei progetti (come sviluppatori e analisti funzionali).

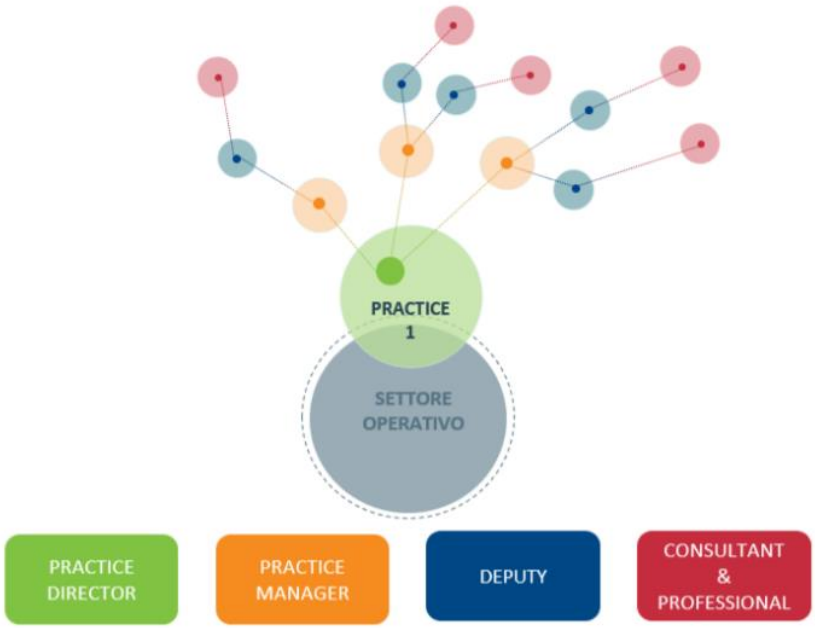


Figura 21 Esempio di organizzazione Techedge S.p.A.

4. ANALISI DEL MODELLO DI GOVERNANCE DI UNA AZIENDA LEADER NEL SETTORE DEI RICAMBI AUTOMOTIVE

4.1 Introduzione

Questo capitolo è volto a illustrare le procedure per la gestione dei servizi IT in una azienda leader nel settore dei ricambi presso il quale la candidata ha avuto modo di collaborare durante l'esperienza formativa di tirocinio.

Lo scopo delle pratiche utilizzate e, in generale della Governance, è quello di amministrare e monitorare l'intero ciclo di vita dei servizi IT in modo tale da ridurre i costi, ma soddisfando i requisiti normativi e di business.

In una realtà aziendale complessa gli attori in gioco sono numerosi: in primo luogo esiste un Cliente che, in questa trattazione, è una azienda leader nel settore dei ricambi automotive.

A livello contrattuale, tale società, ha strategicamente deciso di affidare la gestione della Governance dell'area IT a consulenti esterni, provenienti da diverse società di consulenza. Allo stesso modo, anche i fornitori del servizio (sviluppatori e analisti) sono consulenti esterni che è possibile svincolare al termine del progetto di interesse.

La metodologia implementata per la gestione dei servizi IT è quella ITIL V4.

In particolar modo, l'area interessata nel caso di studio in questione, ha come obiettivo quello di mantenere funzionante e operativo un sistema ERP SAP che gestisce l'approvvigionamento di tutti i mercati NSC (National Sales Company), ovvero tutti quei mercati extra-europei che si riforniscono presso i magazzini centrali presenti in Italia e in America e che, comportandosi come se fossero dei magazzini intermedi, si occupano di rifornire i dealer locali che effettuano gli ordini.

I mercati sono molteplici e situati in tutte le Region del mondo e, per ognuno di essi, il sistema deve tenere traccia del processo di inbound ed outbound dei magazzini coinvolti.

È chiaro che, per poter gestire così tante peculiarità, sia necessario un team corposo di sviluppatori e analisti ed una squadra che si occupi della Governance per mantenere

sotto i controllo i risultati, monitorare eventuali malfunzionamenti, rilevare nuovi cluster su cui poter agire prontamente e mantenere i rapporti con gli utenti di business che, raramente, hanno contatti diretti con i fornitori.

In altri termini, quando si parla di Governance dei progetti, si intende l'anello di congiunzione tra la strategia e il progetto stesso: per fare una metafora, se la strategia indica la direzione da seguire e il progetto è il mezzo per farlo, la Governance ha il compito di fare in modo che non si sbagli strada, comportandosi come il GPS dell'azienda.

Il lato positivo di questa posizione "nel mezzo" è la capacità di colmare le lacune di entrambi i fronti: rispetto alla strategia ha il compito di trovare la giusta dimensione e dare concretezza alla vision aziendale dando supporto alla realizzazione di ciò che non esiste ancora; rispetto al progetto invece ha al suo interno il concetto di time to market e, se è richiesto che un progetto rispetti una milestone prefissata, la Governance deve assicurarsi che, chi si occupa degli aspetti tecnici, sia in linea con le tempistiche di mercato, così da non rendere vano l'investimento economico.

Altre funzioni della Governance sono:

- Cercare di dare un supporto nella gestione del fornitore;
- Mitigare i rischi;
- Prevenire le criticità durante il monitoraggio degli sviluppi: anche se la realizzazione è lasciata in mano al fornitore al 100%, la Governance deve essere aggiornata sulle attività in modo da intervenire prontamente ove necessario;
- Ottimizzare i costi della manutenzione software.

Come premessa per il resto della trattazione, il caso preso in esame tratta di un progetto che ha avuto la sua origine ormai quasi un decennio fa, ma esiste tuttora poiché richiede costantemente numerose modifiche e gap tali da adattare il servizio alle esigenze fluide del mercato e dell'ambiente circostante.

4.2 Il sistema di trouble shooting

Il sistema di trouble shooting, utilizzato dalla Governance per poter essere sempre allineata con tutti i fronti, è nato fondato sui principi della metodologia ITIL V4; tale tool è unico e viene utilizzato per la gestione di tutti i mercati in modo tale da facilitare il lavoro, velocizzarlo ed evitare la burocrazia.

Tale sistema è importante perché:

- Permette una vista centralizzata di tutti i servizi IT;
- Consente una visibilità real time delle segnalazioni fatte dagli utenti;
- Riesce a migliorare la mappatura degli impatti delle diverse modifiche (sia lato ICT, che lato cliente);
- Consente di avere un'unica piattaforma in cui gestire ogni tipo di richiesta: dalla più informatica (come il reset di una password) alla più strategica (come l'implementazione di una nuova funzionalità in grado di aumentare il valore di un'area aziendale);
- Permette a tutti gli utilizzatori di creare e condividere dei report di monitoring.

I 3 moduli principali che lo compongono sono:

1) Incident & Problem Monitoring: per migliorare le procedure attuali, individuando i malfunzionamenti del sistema e tenendo traccia dei corretti KPI;
2) Demand & Change Management: per apportare modifiche (richieste dal cliente) al sistema "As is";
3) Release Management: per definire il calendario delle release, monitorare le modifiche e offrire il giusto supporto a posteriori.

4.2.1 Incident & Problem Monitoring

Un *incident* è definito come una interruzione non pianificata, una riduzione della qualità del servizio o un guasto di un componente ICT che non ha ancora avuto un impatto sul servizio.

Un major incident è un problema prioritario, esso viene aperto solo nel caso in cui il malfunzionamento sia grave o potrebbe causare una interruzione significativa per il cliente.

Un *problem* è la causa di uno o più incidents: esso viene aperto nel caso in cui si sia registrato un malfunzionamento per cui è necessario un ulteriore approfondimento.

In questi frangenti il team di sviluppatori deve agire prontamente per effettuare e condividere una RCA (Root Cause Analysis).

Il ciclo di una vita di un incident si compone delle seguenti fasi:

1. Identification: l'incident è rilevato da utenti interni o esterni;
2. Assigned: l'incident viene assegnato ad un resolver group;
3. Work in progress: questo stato indica che un analista sta lavorando attivamente per la soluzione del problema;
4. Pending: il fornitore del servizio è in attesa di un riscontro da parte di un utente per poter procedere con eventuali implementazioni/modifiche;
5. Escalation: fase facoltativa in cui, nel caso in cui gli sviluppatori non riescano a risolvere il problema, intervengono gli stakeholders di maggiore impatto in modo tale da essere informati del momentaneo malfunzionamento;
6. Resolved: il ticket viene risolto;
7. Closed: dopo 5 giorni dal setting dello stato Resolved, se nessun utente ha ritenuto necessario riaprire l'incident, il ticket viene posto in stato Closed e svanisce la necessità di gestire ulteriormente la casistica.

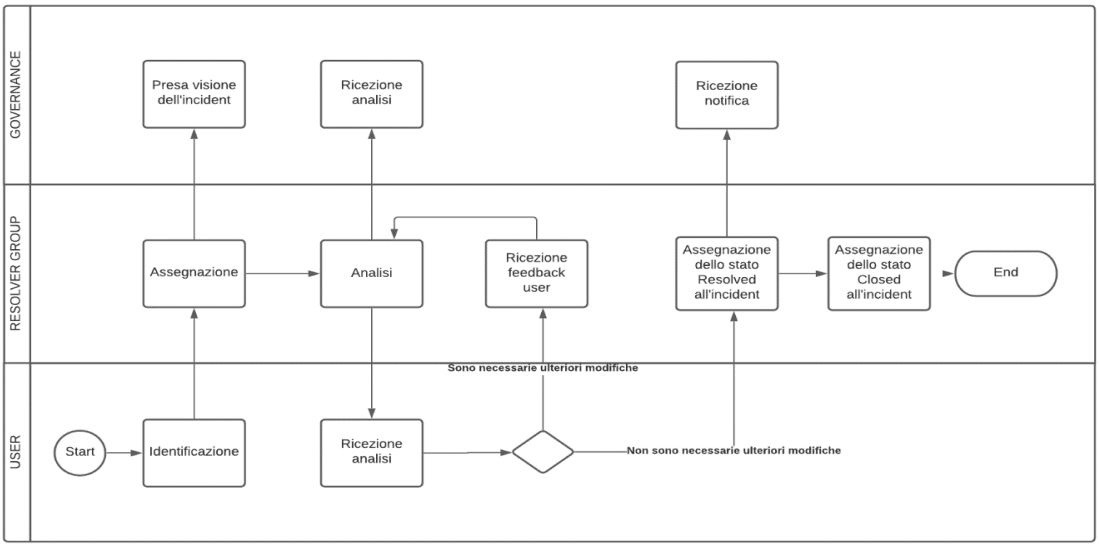


Figura 22 Flow chart Incident

Ad ogni incident viene assegnata dal sistema una severità che determina implicitamente le priorità di risoluzione dei tickets; le severità utilizzate sono le seguenti:

- Severità 3: anomalia poco impattante;
- Severità 2: una funzionalità non sta agendo come richiesto ma è possibile aggirare il problema e/o l'attività che l'utente vuole svolgere non è impattante nell'immediato presente (ma può essere posticipata ad un secondo momento);
- Severità 1: un malfunzionamento sta bloccando le attività del cliente e non è disponibile un workaround, l'utente però può ritardare/posticipare l'attività senza aggravare irreversibilmente lo stato del sistema;
- Severità 0: gli utenti non possono effettuare le attività core di business.
  In questo caso una funzionalità critica non funziona, non ci sono workaround, non è possibile eseguire un riavvio del sistema e l'utente non può posticipare l'azione.

La severità si ottiene a partire dall'impatto e dall'urgenza stabilite dall'utente coinvolto al momento dell'apertura del ticket:

	Urgency High	Urgency Medium	Urgency Low
Impact Very High	0	1	1
Impact High	1	1	2
Impact Medium	2	2	3
Impact Low	3	3	3

Figura 23 Tabella delle severità

In base alla priorità il fornitore deve soddisfare degli SLA (Service Level Agreement) definiti ex ante, in fase di stipulazione del contratto.

Esistono due tipologie di SLA:

- Di response: che riguarda il tempo che intercorre tra l'apertura dell'incident ed il passaggio allo stato Assigned;
- Di resoluton: che riguarda il tempo che intercorre tra l'inizio dello stato Work In Progress ed il passaggio allo stato Resolved.

Per tenere traccia di tutti gli incident aperti è compito della Governance aggiornare costantemente un file Excel di repository che contiene le seguenti informazioni (esempio a solo scopo illustrativo):

Incident	Priority	Market	Status	Opening Date	Short description	Opened by	Affected user
INC001	2	China	Resolved	09/09/2021	Rimozione ordine di vendita errato	F123	F234

La gestione dei Problem, invece, è più complessa e necessita l'intervento di diverse figure:

- Problem Manager: responsabile di valutare i nuovi problemi e decidere se un Problem ha ragione di esistere oppure no;
- Problem Coordinator: persona di riferimento responsabile di comprendere la issue e assegnarla alle persone appropriate all'interno del team (in modo da svolgere ulteriori indagini);
- Problem Owner: responsabile del monitoraggio del ciclo di vita del problema, egli si assicura che il problema venga risolto e che la RCA (Root Cause Analysis) sia adeguata;
- Problem Analyst: esperto in materia all'interno del team di supporto (di sviluppatori), responsabile dell'investigazione del Problem;
- Business Relationship Manager: responsabile della revisione e dell'approvazione del piano di soluzione, egli deve assicurarsi che tutti gli stakeholders siano al correte dello stato della soluzione del Problem.

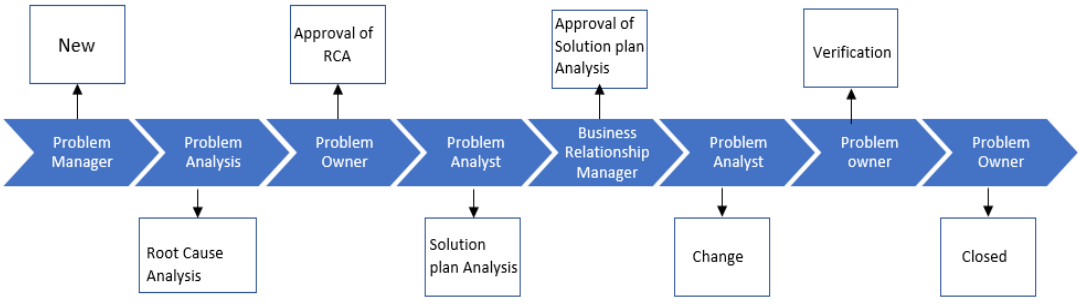


Figura 24 Flow chart Problem

4.2.2 Demand & Change Management

Questa fase è importante per comprendere, anticipare ed influenzare la "domanda" o richiesta del cliente per l'attivazione di nuovi servizi o la modifica di essi; infatti, il sistema utilizzato dagli utenti, non è statico ma, nel corso del tempo, può variare e, a questo proposito, gli utenti possono aprire due tipologie di Demand:

- Enhancement: richiesta di nuovi sviluppi;
- Change: richiesta di attività operative che modificano/migliorano funzionalità già esistenti.

Tali processi devono utilizzare metodi e procedure standardizzate che permettano di registrare tutte le variazioni tenendo sotto osservazione i possibili rischi per il business.

In entrambi i casi sopracitati la richiesta viene posta in stato Draft all'interno del sistema di trouble shooting dallo stakeholder impattato; successivamente la richiesta passa nelle mani della Governance che completa il draft aggiungendo informazioni essenziali affinché il caso passi nelle mani del corretto team di sviluppo:

- Configuration item: qualsiasi componente, applicazione o elemento del servizio che deve essere gestito per assicurare il successo della fornitura;
- Assignment group: team di sviluppatori che prende il carico la richiesta;
- Assigned to: soggetto dell'ICT Governance che acquisisce la responsabilità di approvare le diverse fasi della Demand sia tramite mail che attraverso il sistema di trouble shooting;
- Requested by: utente che ha identificato il caso in esame e che detiene particolare interesse affinché la demand giunga a buon fine.

Una ulteriore differenza tra Change e Enhancement è la seguente: mentre solitamente il Change nasce a fronte di un Incident (e quindi serve per poter migliorare delle funzionalità attualmente affette da bug); gli Enhancement nascono a fronte di nuove richieste riguardanti applicazioni già funzionanti del sistema ma che necessitano dei cambiamenti per andare in contro alle esigenze di business.

A valle di questo si afferma che, salvo eccezioni, i Change sono compresi nel servizio acquistato dal cliente (e la loro spesa in termini di men days è totalmente in carico al fornitore); mentre gli Enhancement devono essere valutati in termini di costi e tempi e,

se approvati, vengono riconosciuti dal cliente come costi on top rispetto al servizio di AMS.

Il processo logico degli Enhancement è illustrato nel flow chart in seguito:

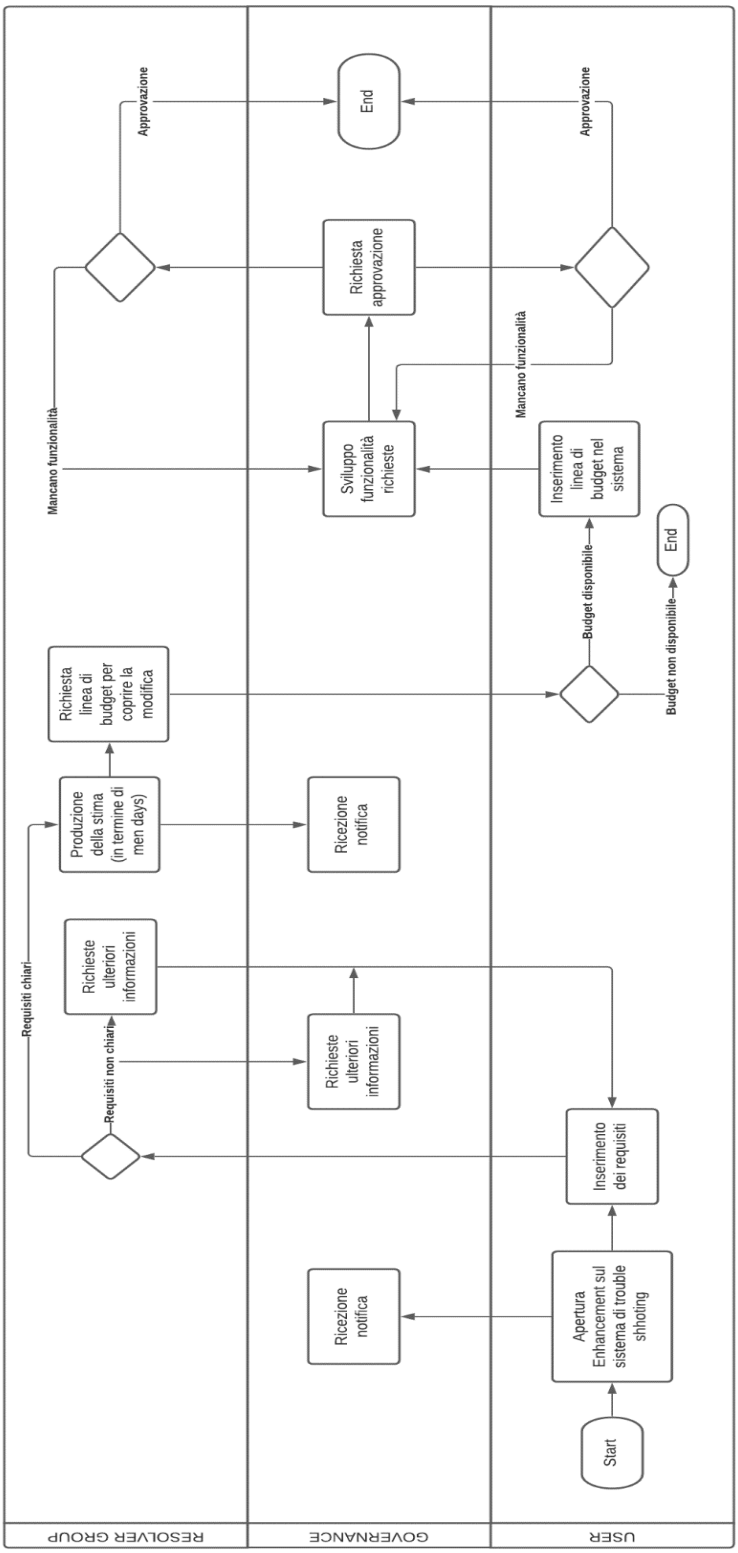


Figura 25 Flow chart Enhancement

4.2.3 Release Management

Il Release and Deployment Management si occupa di gestire le release durante il quale vengono implementati cambiamenti previsti nel Service Design, attraverso la costruzione, il collaudo e la fornitura dei servizi stessi.

Esistono diverse tipologie di release in base alla loro complessità:

- Weekly: una volta alla settimana, servono per portare in produzione piccoli gap o evolutive che possono essere implementate anche a sistema aperto;
- Monthly: una volta al mese (solitamente alle ore 17 durante la giornata di martedì), servono per portare gap più onerosi che, però, possono essere portati a sistema aperto;
- Mayor: una volta ogni quadrimestre; esse si svolgono sempre nel weekend (generalmente a cavallo tra il sabato e la domenica) e necessitano che il sistema sia chiuso per evitare che l'implementazione di nuove funzionalità porti delle complicazioni mentre il sistema è in funzione ed in uso da parte degli utenti.

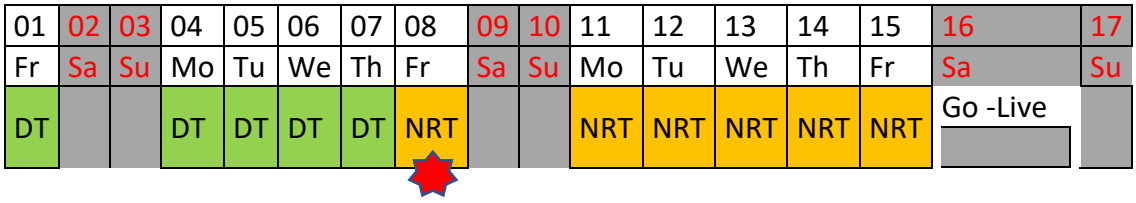
Le fasi del processo di rilascio e gestione della release sono le seguenti:

1. Richiesta di uno o più change e/o funzionalità;
2. Sviluppo delle funzionalità richieste;
3. Fase di test in ambiente di quality;
4. Consegna del CD (Change Document): gli sviluppatori presentano alla Governance e al cliente il CD in cui presentano lo stato "As is" del sistema, confrontato con il "To be" atteso a valle dell'implementazione;
5. Fase di NRT (Non-regression test) in ambiente di quality: questi test sono necessari per verificare se una nuova funzionalità, quando viene messa "a contatto" con altri elementi del sistema, porta a delle regressioni o se, al contrario, riesce ad integrarsi senza portare complicazioni;
6. Release: se la funzionalità in oggetto supera i NRT, può essere portata nell'ambiente di produzione durante la release.

   Durante questa fase di Go-live è importante che il team di sviluppatori tenga sotto costante osservazione il trasferimento in modo tale da poter agire rapidamente in caso di regressioni inattese;

7. Supporto e training per gli utenti coinvolti: viene richiesto un supporto maggiore durante i primi periodi di implementazione delle novità;
8. Revisione e chiusura: verifica se il trasferimento di conoscenze e di formazione è stato adeguato, se tutte le esperienze sono state documentate e tutte le correzioni son state apportate. In caso di esito positivo il processo termina.

Un esempio di calendario di release Mayor è riportato in seguito:



Dove:

DT = Development and Testing;

= Data di consegna dei CD, affinché la modifica possa essere portata durante la Release successiva;

NRT = Non-Regression Test.

4.3 Gestione contrattuale dei fornitori del servizio

4.3.1 Service agreement

I fornitori del servizio ingaggiati dal Cliente costituiscono un Help Desk suddiviso in 2 livelli:

- Primo livello: quando la Governance richiede una richiesta o un ticket da parte di un cliente, e non è in grado di prenderne in mano la gestione autonomamente, deve rivolgersi al team di primo livello.

  Il suo compito è quello di capire le problematiche, proporre soluzioni sia momentanee (workaround) che permanenti (fix), analizzare gli impatti, pianificare i test ed, eventualmente, coinvolgere ulteriori team applicativi.

  Nel caso di studio preso in esame, questo team ha sede in Italia e collabora strettamente sia con la Governance che con il cliente, ove necessario.

- Secondo livello: nel caso in cui l'Help Desk di primo livello non abbia le competenze tecniche necessarie per risolvere un ticket deve girarsi prontamente verso l'Help Desk di secondo livello che ha un compito principalmente di Back Office e sviluppo di software.

  In caso di Incident con severità 0 oppure 1 la Governance deve coinvolgere direttamente il team di secondo livello.

  Tale team ha sede in India e, per vincolo contrattuale, deve garantire una copertura 24 ore su 24, 7 giorni su 7; questa necessità deriva dal fatto che i clienti sono distribuiti in tutto il mondo e, per via dei diversi fusi orari, non sarebbe equo garantire una disponibilità solo parziale.

In entrambi i casi il cliente pattuisce, in fase di progettazione, quale sia il costo giornaliero per ognuna delle risorse e, in base alle commesse che vengono assegnate, si impegna a corrispondere al fornitore l'intero costo delle risorse, indipendentemente dalla bontà di quanto viene consegnato dal fornitore.

Se, ad esempio, il fornitore stima che per un progetto siano necessari 20 men days ad un rate di 280 €/giorno (e se la Governance ritiene tale stima consona), il cliente si impegna a pagare il fornitore, per tale progetto: 20 * 280 = 5.600 €

Tutti i rischi rimangono in capo agli sviluppatori ma, allo stesso modo, tutti gli eventuali savings non vengono spartiti con il cliente; bisogna anche considerare che il rate stipulato tiene in considerazione anche una percentuale di contingency che permetta al fornitore di avere un cuscinetto qualora si verificassero rischi previsti e non.

Lo schema di processo tipico che si verifica in presenza di un malfunzionamento del sistema è il seguente:

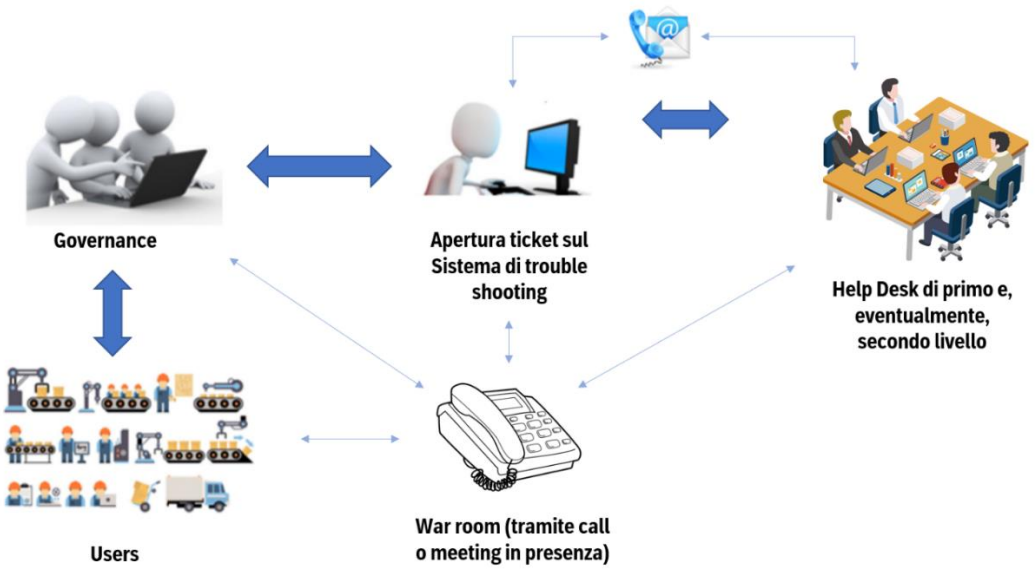


Figura 26 Processo di gestione di un malfunzionamento del sistema

Dove per war room si intende la creazione di un gruppo di lavoro formato dal cliente e dai fornitori per definire le modalità operative con cui gestire, nel minor tempo possibile, il problema emerso.

Essa viene istituita solo in caso di ticket con severità 0 oppure 1 (major incidents) e, in questa sede, è fondamentale determinare chi deve occuparsi della risoluzione del problema e quali sono i tempi di risoluzione.

4.3.2 SLAs

In accordo con le severità illustrate in Figura 23, esistono una serie di KPI vs tempo di risposta che il fornitore accetta in fase di contrattazione.

In seguito è riportata una matrice a scopo illustrativo che mostra, indicativamente, quali sono i tempi di risposta previsti:

KPI (Incident)	Tempo di risposta			
	Severità			
	P0	P1	P2	P3
Presa in carico dell'Incident	*<15(minuti)*	*<15 (minuti)*	*<1 (ora)*	*< 2(ore)*
Workaround (risoluzione del problema puntuale)	*<2 (ore)*	*< 2 (ore)*	*< 3 (ore)*	*< 8 (ore)*
Chiusura dell'Incident (dal momento in cui viene preso in carica)	*< 2 (giorni)*	*< 2 (giorni)*	*< 5 (giorni)*	*< 7 (giorni)*

Figura 27 Matrice KPI vs Tempo di risposta

Ad esempio, il fornitore deve prendere in carico entro 15 minuti dall'apertura, un Incident con severità P1 (indipendentemente dal mercato e dall'orario di apertura).

Per capire se il fornitore ha "bucato gli SLAs" è necessario consultare la seguente ulteriore matrice:

KPI	Rapporto tra il numero di Incident lavorati entro i limiti temporali e il totale degli Incident calcolati su base mensile.			
	Severità			
	P0	P1	P2	P3
Presa in carico dell'Incident	*> 99%*	*> 90%*	*> 80%*	*> 70%*
Workaround	*> 90%*	*> 80%*	*> 70%*	*> 60%*
Chiusura dell'Incident	*> 85%*	*> 85%*	*> 75%*	*> 65%*

Figura 28 Matrice KPI vs Incident lavorati entro i limiti temporali/Incident mensili

Ad esempio, se il fornitore riesce a fornire un workaround, per Incident di severità P1, entro 2 ore l'81% delle volte durante un mese, egli non ha bucato tale SLAs e non deve ricorrere al pagamento di alcuna penale.

Al contrario, ad esempio, se il fornitore riesce a fornire un workaround, per Incident di severità P2, entro 3 ore il 68% delle volte durante un mese, lo SLA viene bucato e sarà necessario provvedere ad un rimborso economico per tale inefficienza del servizio.

In generale, il fornitore deve garantire i livelli di servizio fin dal primo giorno in cui viene ingaggiato; in caso di richiesta di SLAs diversi per il periodo di set-up o per specifici "grace periods", il fornitore deve specificare in dettaglio la strategia che vuole applicare e il cliente deve accettare formalmente la proposta.

In caso di superamento dei valori previsti per i KPI sopraindicati, sono applicate al fornitore delle penali variabili in base all'entità dell'inefficienza.

Ogni mese il cliente comunica al fornitore il mancato rispetto degli SLAs e la penale applicabile (che viene erogata sotto forma di crediti per il cliente); il fornitore può rifiutare la penale con motivazione scritta e, in caso di mancata accettazione della motivazione da parte del cliente, la questione passa nelle mani dello Steering Management.

5. CASO DI STUDIO: APPLICAZIONE DEL MODELLO DEVOPS IN UN CONTESTO AZIENDALE E INTRODUZIONE DEL CONCETTO DI SMART TEAM

5.1 Gestione progettuale Waterfall

Fino ad ora si è discusso del modello di Governance ma, all'interno della realtà aziendale, una parte predominante del tempo impiegato dalle varie risorse è occupata dai progetti.

Per la gestione dei progetti il cliente non utilizza il sistema di trouble shooting introdotto precedentemente, ma utilizza dei tool standard basati sulla metodologia tradizionale Waterfall.

In generale, un progetto IT necessita sia di figure funzionali (analisti) che di figure tecniche (sviluppatori) ma è altrettanto importante la presenza e la collaborazione della Governance che si comporta da collettore tra il fornitore e il cliente.

Inoltre, la Governance, ha la funzione chiave di definire il calendario delle release per fare il modo che le date di rilascio siano compatibili con le esigenze del cliente.

In un progetto Waterfall IT, oltre al cliente e allo sponsor, le figure principali sono:

- Project manager: responsabile della gestione del progetto. Egli, accompagnato dal team, gestisce il progetto applicando un processo di pianificazione e controllo delle risorse tenendo conto di vincoli esterni imposti dal cliente e vincoli interni di tempi e costi;
- Analista: si occupa di analizzare il processo e di preparare tutta la documentazione necessaria. Egli raccoglie i requisiti del cliente, li elabora e li trasforma in un linguaggio tecnico comprensibile agli sviluppatori;
- Sviluppatore: si occupa di comprendere i requisiti tecnici espressi dal funzionale e di tradurli in linguaggio di programmazione fino ad ottenere la funzionalità attesa dal cliente.

In un mondo teorico l'analista, dopo aver preparato l'analisi funzionale a partire dai requisiti raccolti dal cliente, può essere svincolato dal progetto per poi tornare ad essere coinvolto in procinto della release per la preparazione dei casi di test.

I casi di test sono svolti dagli sviluppatori ma, al termine di questa fase interna, è compito degli analisti coinvolgere il cliente durante l'UAT per ricevere l'approvazione al collaudo finale.

L'UAT (User Acceptance Testing), è definito come il test del software da parte dell'utente per determinare se esso può essere accettato (e quindi passare al rilascio in produzione) o meno.

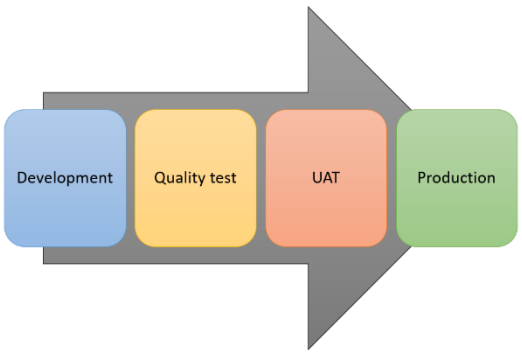


Figura 29 Fasi dello sviluppo del software

Se viene preso in considerazione l'asse temporale di un progetto Waterfall (Figura 30) è possibile identificare delle inefficienze:

		2021											
		Ott					Nov				Dic		
		W39	W40	W41	W42	W43	W44	W45	W46	W47	W48	W49	W50
Progetto	Kick-off												
	Analisi funzionale												
	Sviluppo software/Test interni												
	Preparazione casi di test												
	UAT												
	Test di non regressione												
	Go Live												

Figura 30 Asse temporale progetto Waterfall

Legenda:

	Analista
	Sviluppatore

Durante la fase di sviluppo l'analista non ha alcuna mansione e, viceversa, durante la fase di analisi e di preparazione dei casi di test, il team di sviluppo non viene coinvolto.

Fatta questa premessa, diviene problematico per il Project Manager allocare al meglio le sue risorse perché, se in un mondo ideale è possibile per gli elementi del team dedicarsi a più progetti contemporaneamente in modo da riempire i tempi "di attesa", nella realtà è un processo articolato tale da rendere gli switching cost di questa operazione spesso elevati.

Questo tipo di gestione progettuale, in caso di mancato rispetto delle milestone, comporta problemi anche lato cliente: in queste circostanze, infatti, si genera un over cost da gestire. Spesso però è immediato identificare quale delle due parti debba accollarsi il costo: di norma è sostenuto dal fornitore nel caso in cui non abbia svolto correttamente il suo lavoro, oppure dal cliente se i ritardi sono riconducibili a sue mancanze; questo confine è molto sottile e può provocare conflitti tra le parti coinvolte.

5.2 Gestione progettuale DevOps – Smart Team

A partire da 2019 è stato coniato internamente dall'azienda Techedge S.p.A., il concetto di Smart Team che si basa sui fondamenti del modello DevOps.

All'interno dello Smart Team non sono più presenti le figure elencate precedentemente in ambito Waterfall ma nascono degli specialisti con competenze trasversali in grado di svolgere sia compiti funzionali sia compiti di natura tecnica: tali figure diventano anche dei deputy Project Manager poiché gestiscono, con un alto grado di autonomia, il proprio tempo e sono di riferimento per le figure più tecniche.

Come conseguenza, grazie allo Smart Team, il PM delega molti task di connotazione progettuale al team ed interviene solo per effettuare dei monitoraggi periodici e assicurarsi che le risorse non siano in overhead; la pianificazione delle attività più tecniche, invece, passa direttamente allo Smart Team che, in questo modo, si responsabilizza tramite l'opportunità di affrontare esperienze al di fuori della propria confort zone tradizionale.

Questo concetto si collega bene al CSI trattato nel capitolo 1.2.2: uscire dalla routine consente anche alle persone più tecniche di adottare un approccio nuovo orientato verso attività di R&D.

Uno dei principi dello Smart Team, infatti, spinge le persone ad essere responsabili delle loro azioni e, secondo una logica di continuos improvement, le porta a stimolare la voglia di conoscenza in tutte le circostanze in cui si manifestano lacune tecniche e/o concettuali.

Questo atteggiamento è tipico di un progetto Agile – DevOps mentre non si adatta bene ad un progetto Waterfall in cui si assume che gli stakeholders di progetto sappiano fin dal principio cosa fare e che siano in grado di rispettare milestone chiare e ben definite.

In assenza di Smart Team, quindi, i ruoli all'interno dell'organizzazione del fornitore del servizio sono ben distinti agli occhi del cliente:

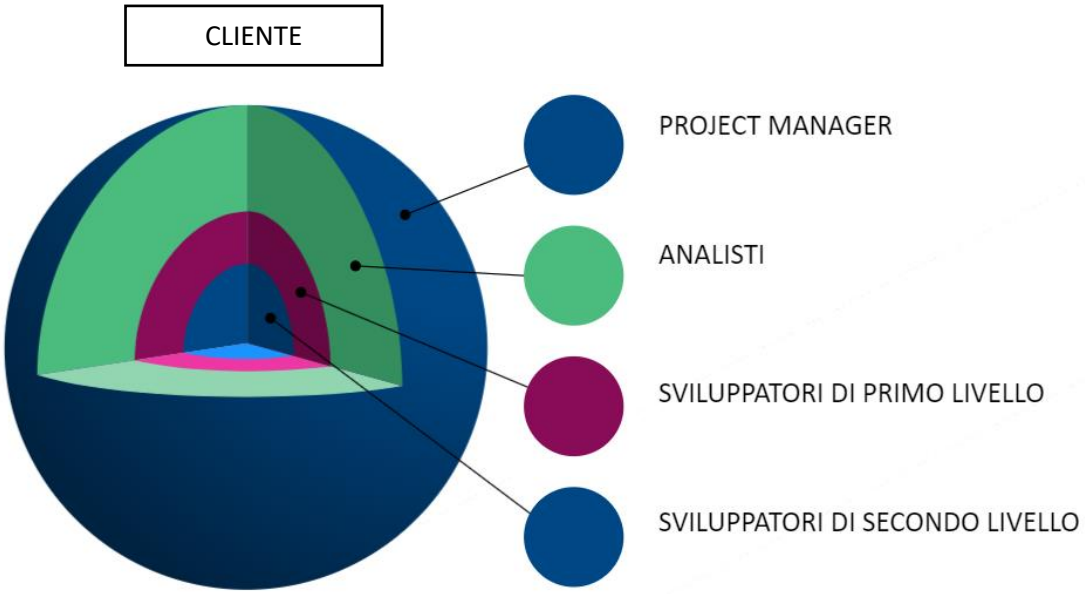


Figura 31 Organizzazione Waterfall

Il cliente, in prima battuta, si interfaccia con il Project Manager; tramite il PM il cliente si interfaccia (in un secondo momento) con gli analisti che, a loro volta, hanno il compito di mantenere allineati gli sviluppatori sia di primo che di secondo livello (vedere Capitolo 4.3.1).

In presenza di Smart Team, invece, i confini divengono meno netti: si mantiene un nucleo concentrato di tecnici che si occupano unicamente della parte di sviluppo (secondo livello) ma, d'altra parte, nasce una figura che ha la capacità di fondere diverse professionalità con l'obiettivo di affrontare ogni necessità.

Si introduce così il concetto di professionalità orizzontale:

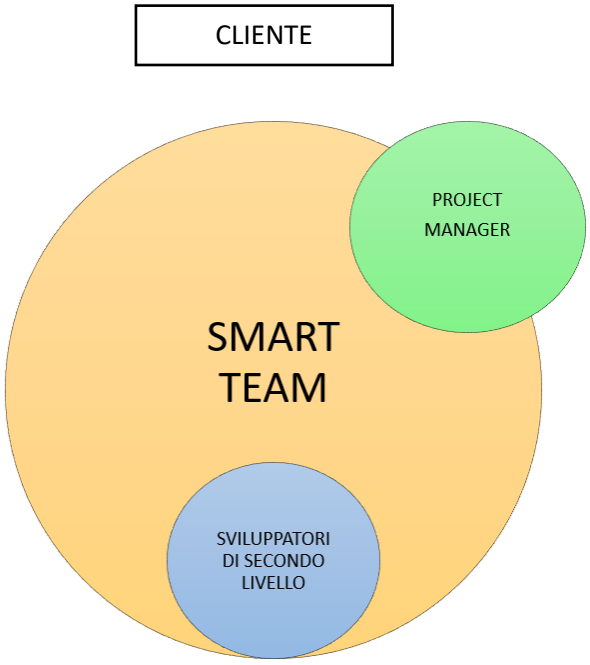


Figura 32 Organizzazione DevOps

Utilizzando lo Smart Team, quindi, si passa dal commissionare interi progetti al commissionare microservizi e, sotto il punto di vista contrattuale, il cliente non paga un progetto chiavi in mano ma provvede al pagamento delle singole professionalità.

In modalità Waterfall il cliente acquista un progetto andando a specificare delle milestone durante la fase di inizializzazione e identificando, in fase di kick-off meeting, quali saranno i requisiti da sviluppare (con le relative priorità, seguendo un diagramma di Gantt). In caso di necessità di nuovi requisiti in corso d'opera, il cliente deve richiedere una CR (Change Request) che comporta una ulteriore spesa in termini di man days oltre alla baseline pianificata.

Al contrario, se il cliente ingaggia uno Smart Team, è consapevole di poter usufruire di uno staff di persone con background variegato che è in grado di adattarsi a differenti

evenienze. In questo modo il cliente non deve pianificare al dettaglio l'intero progetto nelle sue fasi iniziali, ma deve solamente fornire una lista dettagliata delle attività in pipeline. In corso d'opera le priorità possono cambiare e, con lo Smart Team, l'ordine delle attività può essere facilmente variato e possono essere aggiunti o modificati task senza alcuna spesa ulteriore perché, come già specificato, il cliente non compra un progetto ma un servizio.

Per comprendere il concetto viene riportato un caso di sviluppo di un'applicazione:

- Progetto Waterfall: se in fase di pianificazione il cliente richiede che l'applicazione non debba possedere un determinato report, se in un secondo momento cambia idea egli dovrà pagare per la richiesta effettuata (assumendo che sia ancora possibile tecnicamente procedere alla modifica);
- Progetto DevOps: se il cliente ritiene, in corso d'opera, che sia necessaria l'introduzione di un nuovo report (seppur tale task non fosse presente nella pipeline iniziale delle attività), assumendo che non sia troppo tardi per effettuare tale modifica, lo Smart Team deve prestarsi alla modifica senza richiedere alcun pagamento aggiuntivo al cliente.

Il cliente, contrattualmente, si impegna a pagare le risorse per un determinato periodo e, entro i limiti del periodo indicato, può chiedere richiedere tutte le attività/modifiche che ritiene necessarie.

D'altra parte, siccome il cliente richiede un servizio si espone ad un rischio; ovvero che le risorse previste non siano sufficientemente alimentate di lavoro: risorse mediamente scariche che devono ugualmente essere pagate.

In ottica di microservizi, però, è molto improbabile che si verifichi questo scenario.

In generale, è responsabilità del cliente gestire le attività infatti è lui che si occupa di definire le priorità e talvolta, strategicamente, potrebbe preferire avere un Project [1]Earned Value più basso in modo tale da dare la precedenza ad attività che stima possano portare valore aggiunto all'organizzazione.

[1] L'Earned Value EV è il valore a budget del lavoro eseguito in determinato periodo e si calcola come:
EV = Budget Cost*Work Performed

In ambito Waterfall, invece, non è possibile intraprendere scelte di questo genere; la storia però insegna che i contesti cambiano nel tempo e si possono generare bisogni che, in fase di inizializzazione, non si sarebbero potuti immaginare.

Uno dei principali cambiamenti imprevedibili che ha stravolto i paradigmi è certamente l'epidemia di Covid-19: il cliente sempre di più si accorge che le sue priorità sono mutevoli e fortemente influenzate dalla digitalizzazione.

5.3 SWOT Analysis

5.3.1 SWOT Analysis lato cliente

La SWOT (Strenghts, Weaknesses, Opportunities and Threats) Analysis è una metodologia utilizzata per effettuare scelte strategiche a partire dalla mappa dei fattori interni ed esterni, positivi e negativi, di una organizzazione.

Figura 33 SWOT Analysis (lato cliente)

1. Strenghts: da un punto di vista contrattuale, c'è continuità sulle risorse all'interno del progetto. Inoltre, le milestone sono meno stringenti infatti, come specificato in precedenza, le priorità possono cambiare e quindi anche i traguardi divengono più flessibili;

2. Weaknesses: c'è il rischio di non alimentare correttamente delle risorse che il cliente è vincolato a pagare per il periodo di tempo pattuito in fase di procurement;
3. Opportunities: il cliente non si impegna su un progetto ma sulle risorse, quindi possono cambiare le priorità in corso d'opera.

   Inoltre, siccome le risorse sono ibride, il cliente ha diversi punti di riferimento all'interno del team aumentando così le possibilità di comunicazione: coinvolgendo più figure è più semplice parallelizzare le attività senza creare colli di bottiglia;
4. Threats: c'è una fluidità nelle responsabilità delle varie attività. Le risorse non hanno un perimetro chiaro di azione e quindi potrebbe divenire difficile tenere sotto controllo il modello: in caso di problemi non è semplice capire come e su quale risorsa intervenire per risolverli.

5.3.2 SWOT Analysis lato fornitore

Da punto di vista del fornitore la SWOT Analysis è la seguente:

Figura 34 SWOT Analysis (lato fonitore)

1. Strenghts: il fornitore può ottimizzare le risorse sui progetti garantendo loro una formazione professionale adatta a questo tipo di modello. Inoltre, uno dei principali punti di forza riguarda la copertura economica: il cliente si impegna per

pagare lo Smart Team per un periodo temporale che spesso è più alto rispetto alla durata prevista del progetto di interesse; in questo modo il fornitore acquisisce una copertura su un più lungo orizzonte e diviene più interessato nell'investire sulle risorse in modo tale da plasmarle in base alle esigenze del cliente;

2. Weaknesses: competenza tecnica trasversale che potrebbe definire dei profili unici e non reperibili sul mercato, con il rischio di non poter integrare le risorse;
3. Oppotunities: è possibile gestire una maggiore scalabilità verso nuovi progetti;
4. Threats: la specializzazione, specie della componente funzionale sul cliente, è sicuramente un punto di forza mentre la commessa è in corso ma si presenta il rischio che, al termine dell'ingaggio, le risorse siano fortemente specializzate e, in caso di mancato rinnovo della commessa, potrebbe essere difficile riallocarle.

5.4 Differenza tra Smart Team e Team Agile

Da un punto di vista concettuale, lo Smart Team DevOps ed il Team Agile che segue la metodologia Scrum, sono molto simili.

La differenza è principalmente contrattuale: seguendo le pratiche Agile, infatti, si mantiene un concetto di delivery del fornitore presso il cliente ed il contratto tra le parti pone la sua attenzione quasi completamente sul prodotto/servizio erogato. Utilizzando uno Smart Team, invece, si genera una sorta di Partnership tra fornitore e cliente.

In ottica DevOps il cliente, nel momento in cui si lega contrattualmente al team, riceve (incluso nel pacchetto) anche il servizio di Service Management erogato dal PM alla guida del team; se il cliente invece opta per un contratto di delivery (che sia Agile o Waterfall) deve provvedere autonomamente a fornire un PM interno.

Riassumendo, mentre con uno Smart Team il PM del fornitore è una figura di riferimento anche per il cliente, in ottica Scrum ciò non è ammissibile e la conseguenza è una maggiore spesa sia economica che in termini di tempo per il cliente.

Un meccanismo del genere, però, è funzionante solo se tra le parti c'è un rapporto di stima reciproca; per questo motivo potrebbe essere complesso implementare uno

Smart Team in un progetto emergente ma, tale scelta, potrebbe essere ottimale se adottata in una fase più matura del progetto, dopo il raggiungimento di un livello di fiducia elevato, in cui il sistema è collaudato ed il fornitore ha già acquisito delle competenze specifiche.

Il cliente, in questo modo, delega molta responsabilità ma, d'altro canto, ha un risparmio poiché non deve sostenere i costi di una sua risorsa dedicata a fare il PM sul progetto.

6. CONCLUSIONI

In questo capitolo si puntualizzeranno i possibili sviluppi futuri, con i relativi vantaggi e i limiti riscontrati, e si identificheranno i benefici della tesi e del tirocinio svolto.

6.1 Possibile applicazione futura

Tra i possibili sviluppi futuri vi è senza dubbio l'applicazione del modello DevOps al modello attualmente utilizzato dal cliente presso il quale è stato erogato il servizio.

Per poter cogliere i benefici di tale implementazione, applichiamo il modello DevOps alla gestione AMS (Application Management Services): per le aziende che utilizzano sistemi avanzati come SAP, la sfida principale di ogni giorno è quella di evitare, o risolvere prontamente, eventuali problemi tecnici legati all'operatività applicativa. Per non rallentare i processi, quindi, diviene essenziale poter contare su un servizio efficiente di AMS al fine di poter risolvere, in tempi rapidi, eventuali anomalie legate all'utilizzo del sistema.

Il cliente paga, attualmente, 2 risorse di primo livello e 4 risorse di secondo livello per gestire l'AMS dell'ERP; per la seguente analisi si focalizzi l'attenzione soltanto sull' Help Desk di primo livello.

Si riportino in seguito il grafico e la tabella che illustrano il numero di ticket mensili che devono essere risolti dall'Help Desk di primo livello (considerando in aggregato tutti i mercati gestiti dal sistema):

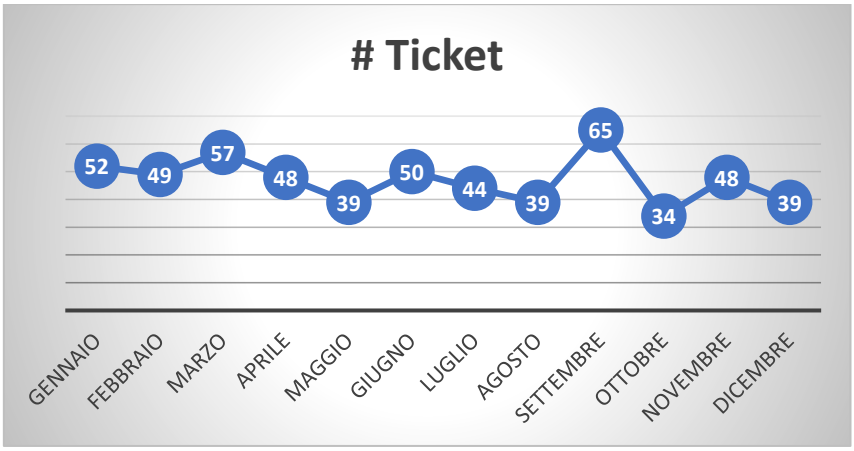


Figura 35 Grafico andamento ticket Help Desk di primo livello

Mese	# Ticket
Gennaio	52
Febbraio	49
Marzo	57
Aprile	48
Maggio	39
Giugno	50
Luglio	44
Agosto	39
Settembre	65
Ottobre	34
Novembre	48
Dicembre	39
Media	**47,00**
Dev. Standard	**8,71**

Quando il cliente deve decidere, all'inizio dell'anno contabile, quante risorse allocare come AMS di primo livello, visualizza il trend dell'anno precedente, calcola il numero medio di ticket di competenza e la relativa deviazione standard.

Si noti come la deviazione standard sia molto elevata; questo è sinonimo di un contesto non ancora collaudato in maniera ottimale. Se la deviazione standard fosse bassa, al contrario, il cliente avrebbe una stabilità mensile dei ticket e non avrebbe motivo di andare ad adoperare dei modelli di business più innovativi, diversi da quello tradizionale.

In un secondo momento viene consultata anche la seguente tabella, che nasce sulla base di serie storiche:

Difficoltà di risoluzione	Tempo di risoluzione medio	# medio ticket (per anno)	%
Bassa	2 h	264	47%
Media	6 h	185	33%
Alta	10 h	115	20%

Tempo medio annuo impiegato nella risoluzione di ticket = 264 * 2h + 185 * 6h + 115 * 10h = 2788 ore all'anno circa

Tenendo conto che ogni risorsa eroga 8 ore giornaliere e considerando circa 220 giorni lavorativi annui:

Ore lavorative totali (per una risorsa) = 220 * 8h = 1760 ore all'anno

Risorse necessarie = 2788h / 1760h = 1,58 risorse

Il cliente, a valle di questi calcoli, può decidere di vincolarsi contrattualmente a 2 risorse di AMS di primo livello per l'anno successivo.

In aggiunta a questo servizio il cliente ha a disposizione un plafond per i Change, ovvero un numero massimo di ore che il team, da contratto, può erogare per la gestione delle piccole evolutive (i progetti sono esclusi).

Il plafond, nel caso di studio di riferimento, è di 720 ore annue che devono essere distribuite in maniera uniforme nei 12 mesi dell'anno (60h/mese).

Teoricamente, acquisendo 2 risorse:

Ore totali a disposizione = 2 * 8h * 220 giorni = 3520 ore all'anno

Di conseguenza, le ore che ipoteticamente rimangono libere per lo sviluppo e l'implementazione delle evolutive sono: 3520h – 2788h = 732 ore all'anno.

Siccome tale valore è maggiore rispetto a 720, è atteso che il team, mediamente, riesca a gestire il carico di lavoro assegnato; le problematiche però sorgono in caso di picchi inattesi di ticket o, viceversa, in caso di carenza di ticket dovuta, ad esempio, a festività comuni in diversi Paesi di competenza.

In questi ambiti, il vincolo delle 60 ore mensili disponibili per lavorare sui Change, potrebbe diventare stringente:

- Nei mesi in cui i ticket aperti sono sopra la media potrebbero essere necessarie più di 2 risorse per non generare backlog. Ad esempio, si analizzi nel dettaglio il mese di settembre:

Difficoltà di risoluzione	# medio ticket (settembre)
Bassa	32
Media	21
Alta	12

  Le ore medie necessarie per la risoluzione dei ticket sono: 32 * 2h + 21 * 6h + 12 * 10h = 310 ore al mese

Le ore erogabili in un mese dalle due risorse (straordinari esclusi) sono: 2 * 8h * (220/12) giorni = 294 ore al mese, arrotondando per eccesso.

Quindi, se non si vuole generare un backlog molto elevato di ticket, le attenzioni del team devono focalizzarsi totalmente sulla risoluzione di problemi e viene completamente persa l'occasione di sfruttare le 60 ore a disposizione per le evolutive.

- Al contrario, nei mesi in cui il numero di ticket è notevolmente sotto la media, le due risorse avrebbero a disposizione anche più di 60 ore per lavorare sulle evolutive ma, siccome esse non sono accumulabili nel tempo, non riescono a sfruttare a pieno il proprio tempo rimanendo mediamente scariche.

  Si analizzi nel dettaglio il mese di ottobre:

Difficoltà di risoluzione	# ticket (ottobre)
Bassa	16
Media	12
Alta	6

  Le ore medie necessarie per la risoluzione dei ticket sono: 16 * 2h + 12 * 6h + 6 * 10h = 164 ore al mese

  Le ore erogabili in un mese sono 294 perciò le ore che potrebbero essere utilizzate per le evolutive sono 130 ma, siccome da contratto il limite massimo mensile è di 60 ore, circa 70 ore mensili vengono sprecate.

Il risultato che emerge è che nei mesi in cui il numero di ticket è elevato, la gestione di entrambi i servizi (incident ed evolutive) viene svolta spesso in modo superficiale per poter portare a termine più task possibili nel minimo tempo mentre, nei mesi con basso numero di incident, i task vengono svolti con maggiore attenzione dal team ma il cliente è costretto a pagare delle risorse mediamente scariche che non investono in maniera ottimale il proprio tempo.

La situazione mensile media delle ore si può riassumere nella tabella in seguito:

	Gen	Feb	Mar	Apr	Mag	Giu	Lug	Ago	Set	Ott	Nov	Dic
HD	256	242	278	236	194	244	216	194	310	164	236	194
Plafond	60	60	60	60	60	60	60	60	60	60	60	60

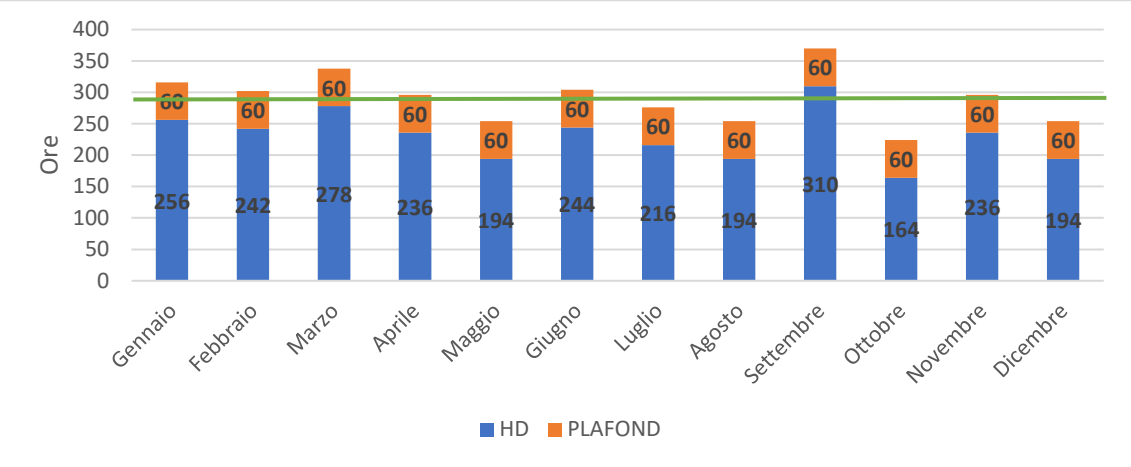


Figura 36 Distribuzione mensile HD-Plafond

Fissando il limite delle 294 ore mensili, dal grafico in Figura 36, risulta evidente che gli unici mesi vicini alla situazione ottimale sono Aprile e Novembre; per cinque mesi le risorse sono sovraccaricate mentre, per i restanti mesi, si verifica uno spreco.

Implementando uno Smart Team, invece, svanisce il vincolo di contratto sul plafond delle evolutive: il cliente ha la possibilità di acquisire 2 risorse ibride (il più possibile intercambiabili) che gestiscono autonomamente il proprio tempo dedicandosi ogni mese ai ticket e ai Change in quota variabile, in base alle necessità.

Anche in questo caso, quindi, avere una maggiore libertà nella gestione delle priorità, permette di organizzare in modo ottimale il carico di lavoro mensile (e l'eventuale backlog dei mesi precedenti).

Partendo dal presupposto che, anche in questo scenario, le ore mensili sfruttabili dalle risorse siano 294:

	Gen	Feb	Mar	Apr	Mag	Giu	Lug	Ago	Set	Ott	Nov	Dic
HD	256	242	278	236	194	244	216	194	310	164	236	194
Plafond	38	52	16	58	100	50	78	100	-16	130	58	100

Totale ore spendibili per le evolutive = $\sum Plafond$ = 764 ore all'anno

Senza l'implementazione di DevOps, invece, potevano essere sfruttate per i Change al massimo 720 ore all'anno quindi, anche nel caso puramente teorico in cui fosse possibile sfruttare tutte e 720 ore a disposizione, si verifica una perdita di circa 44 ore.

Percentuale di tempo guadagnato sulle evolutive implementando il modello DevOps = 44h / 720h = 6,1%

Come già citato nel Capitolo 5.3.1, però, esiste la minaccia che le responsabilità siano eccessivamente fluide e che il perimetro di azione mensile non sia sempre chiaro e visibile agli occhi del cliente: in tali casi, quest'ultimo, potrebbe perdere il controllo del modello.

6.2 Analisi quantitativa del servizio

6.2.1 Analisi degli SLAs

Da un punto di vista quantitativo, per capire se il sistema funziona efficientemente, è possibile considerare una serie di indicatori legati agli SLAs.

Attualmente, per il monitoraggio ed il controllo degli SLA, viene utilizzata la matrice in Figura 27: i vincoli temporali indicati, però, non sempre riescono ad essere rispettati dal fornitore del servizio poiché è usuale la presenza di ticket con severità P1-P2 che necessitano una presa in carico ed un workaround molto rapidi, indipendentemente dalla difficoltà di risoluzione del ticket.

In ottica Waterfall, inoltre, si aggiunge la complicazione di dover spesso coinvolgere diverse figure: siccome le risorse non sono ibride, è necessario un dialogo tra Help Desk

di primo e secondo livello e, nei casi più complessi, può anche essere inevitabile l'intervento di un analista che sia in grado di andare a coprire alcune lacune strettamente funzionali legate al singolo incident.

È chiaro che l'intervento di più persone possa essere causa di colli di bottiglia e possa rallentare ampiamente i tempi di lavoro: se anche solo una figura chiave non può fornire il suo supporto, il ticket rimane in sospeso andando a "bucare gli SLAs".

Implementando uno Smart Team, invece, è più difficile che si presenti una situazione del genere perché ogni risorsa è teoricamente in grado di gestire autonomamente i ticket: questo riduce le tempistiche in quanto vengono quasi totalmente eliminati i tempi di attesa di risposta tra un interlocutore ed un altro.

Siccome il team, presso il quale è stato svolto il tirocinio, non è organizzato tramite Smart Team, però, non è possibile mostrare delle evidenze numeriche di quanto sopra sostenuto.

Si possono però identificare alcuni KPI utili per tracciare aspetti legati agli SLA come, ad esempio:

1. % di ticket che non rispettano gli SLA per bucket temporale e tipologia;
2. SLA medio per tipologia di ticket e deviazione standard;

Inoltre, a partire dalla distribuzione dei ticket mensili e dal numero di SLA bucati nel periodo, è possibile valutare la prestazione del fornitore.

Le casistiche che si possono presentare sono le seguenti:

- Il fornitore eccede gli SLA in maniera frequente, indipendentemente dalla distribuzione dei ticket: in questa situazione è probabile che esso abbia dei problemi di competenza tecnica o che l'applicazione abbia dei bug importanti difficili da risolvere;
- Il fornitore buca gli SLA solo nei periodi in cui si manifestano dei picchi di ticket: in questo caso, al contrario, è probabile che non ci siano carenze conoscitive piuttosto legate al numero di risorse presenti nel team.

6.2.2 Ticket Intelligence e RC Analysis

Un altro frangente in cui sarebbe possibile dimostrare un miglioramento quantitativo è quello legato al concetto di Ticket Intelligence: periodicamente, infatti, le risorse di primo livello sono chiamate a svolgere una analisi approfondita dei ticket in modo tale da andare ad individuare le cause radice più ricorrenti.

Si ipotizzi di effettuare tale Ticket Intelligence su un intero anno: dato un totale di 565 incident annuali, escludendo circa 200 ticket per cui non è possibile individuare una causa radice (ad esempio, poiché si tratta di errori manuali dell'utente o di False Problem), per i rimanenti vengono individuate le seguenti RC:

Cluster di RC	Numerosità annuale
Fallimento trasmissioni dati da altri sistemi	125
Errata creazione ordine di vendita	83
Disallineamenti anagrafici	51
Altro	44
Mancato invio dati ad altri sistemi	32
Blocco fisico del materiale in un punto della supply chain	23
Time-out di rete	7

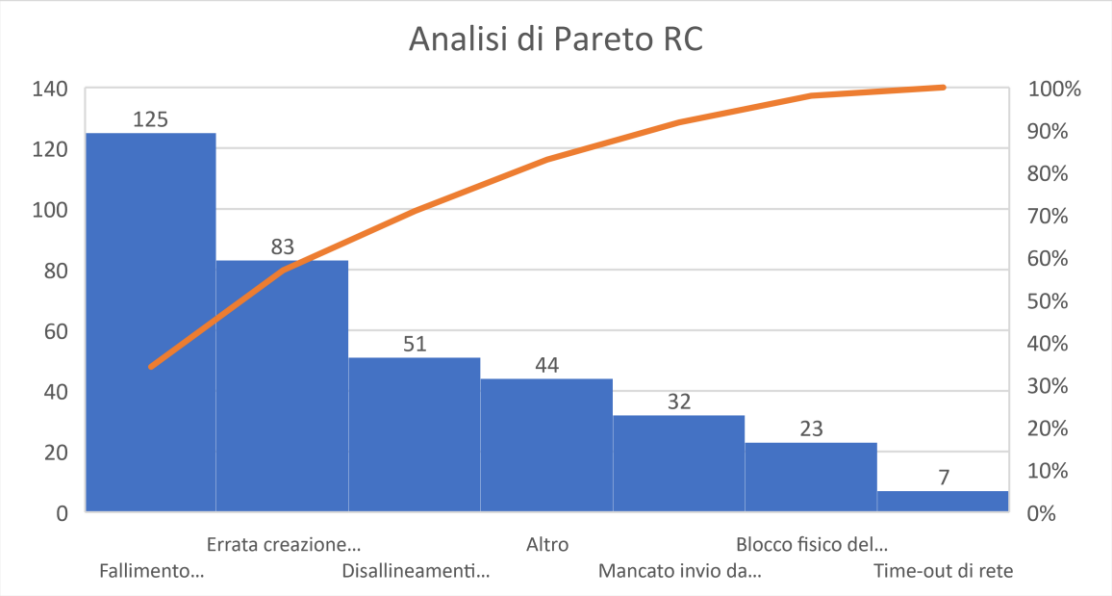


Figura 37 Analisi di Pareto delle RC

Questa analisi consente di capire graficamente ed in maniera immediata quali sono le RC che impattano il maggior numero di ticket: su di esse vale la pena focalizzare l'attenzione in modo prioritario andando a proporre delle soluzioni volte al miglioramento definitivo delle funzioni impattate, evitando l'apertura di ticket analoghi in futuro. Si noti che diverse tipologie di incident, seppur apparentemente scollegate tra loro, possono avere una causa comune.

Si noti, inoltre, che le ore in cui vengono effettuate tali RCA, devono essere considerate come ore di evolutive.

Si prenda come esempio il cluster di RC più numeroso "Fallimento trasmissioni dati da altri sistemi": le risorse, per eliminare tale causa radice, devono capire la motivazione del fallimento (che può essere, ad esempio, dovuto a micro-interruzioni di rete) e devono andare ad intervenire su una serie di configurazioni in modo tale da ridurre significativamente i casi di insuccesso.

Per poter effettuare tale studio approfondito sia stima l'HD1 necessiti 20 ore totali.

In regime Waterfall, per poter rimanere all'interno dei limiti di Plafond portando a compimento anche gli altri Change in pipeline, si stima che tale analisi possa essere terminata in circa 4 mesi mentre, utilizzando uno Smart Team, siccome tale vincolo di Plafond viene a mancare, il tempo per concludere il task si riduce ampiamente.

Tale meccanismo è virtuoso e permette di godere dei benefici della RCA rapidamente, limitando le rigidità imposte dal modello ITIL e generando un loop miglioramento sotto due punti di vista:

- Riflesso quantitativo: diminuzione del numero di ticket e possibilità di interfacciarsi con un maggior numero di cluster, a parità di tempo disponibile;
- Riflesso qualitativo: il cliente nota un comportamento proattivo del team e la sua soddisfazione nei confronti del servizio erogato aumenta.

6.3 Benefici del tirocinio e della tesi

Il tirocinio svolto, seppur nelle difficoltà dovute al particolare momento vissuto per causa del Covid-19, è stato comunque coinvolgente e, in quanto correlato alla tesi di

laurea, ha contribuito a sviluppare una partecipazione ancora più attiva e consapevole permettendo di comprendere l'importanza e la complessità del settore IT all'interno delle aziende.

Questo lavoro di tesi parte illustrando il modello attualmente utilizzato dal cliente sostenendo che il Framework ITIL V4, se gestito correttamente, è funzionale ed estremamente prezioso per il mondo IT.

Questo modello, però, non è l'unico possibile che può essere seguito dalle imprese che erogano servizi IT: il metodo DevOps si dimostra all'avanguardia e cerca di stravolgere alcune logiche del Project Management classico ponendosi come uno dei principali candidati per diventare essenziale, in un futuro prossimo, nella soluzione di problemi e nello sviluppo di nuove funzionalità software.

Nel caso in esame ci si è focalizzati sul modello DevOps analizzando soprattutto il concetto di Smart Team, per capire come una gestione più fluida delle risorse e dei task possa generare valore ma, anche in ambito di programmazione, il modello sta ottenendo ottimi risultati.

Da uno studio condotto nel 2019 emerge però che il modello è ancora accettato solo parzialmente dalle aziende infatti, in Italia, il 90% delle organizzazioni rischia di fallire nell'approccio DevOps: questo accade perché, passare a tale framework, implica un vero e proprio cambiamento culturale e le aziende sono spesso intimorite nel momento in cui provano a sostituire i loro processi di sviluppo tradizionali.

Le cause principali di questo atteggiamento sono due:

1. Timore di dover sostenere un costo troppo alto per attuare il cambiamento;
2. Rischio di non ottenere il successo sperato in quanto il modello non è ancora ben radicato nella cultura aziendale e presenta rischi (ma anche opportunità) maggiori rispetto ad un modello classico Waterfall.

Nonostante questi timori però alcune aziende, in particolar modo le startup, stanno sempre più promuovendo l'uso di questa metodologia: si ipotizza che tali organizzazioni possano diventare il veicolo delle nuove tecnologie in ambito Digital Transformation e

che, tramite un passaggio lento e graduale nel tempo, possano essere in grado di influenzare e coinvolgere anche le Large Enterprises.

SITOGRAFIA

- https://www.itpro.co.uk/business-operations/31711/what-is-a-managed-it-service
- https://blog.itil.org/2013/01/from-project-portfolio-to-service-portfolio-management/
- https://www.cmdbuild.org/it/file/pmbok-iv-e-itil-v.3
- https://www.axelos.com/best-practice-solutions/itil/what-is-it-service-management
- https://www.qrpinternational.it/blog/glossario/service-manager-chi-e-e-cosa-fa/
- https://overneteducation.it/Home/Approfondimento/70/it-service-management-e-it-governance
- https://it.wikipedia.org/wiki/ITIL
- https://vitolavecchia.altervista.org/che-cose-e-i-vantaggi-itil-in-azienda/
- https://info.axiossystems.com/blog/what-are-the-four-dimensions-of-itil-4
- https://worldofagile.com/blog/four-dimensions-of-service-management-in-itil4/
- https://www.bmc.com/blogs/itil-four-dimensions-service-management/
- https://www.bmc.com/blogs/itil-service-value-system/
- https://www.qrpinternational.it/blog/it-governance-and-service-management/i-7-principi-guida-itil/
- https://www.bmc.com/blogs/itil-governance/
- https://www.bmc.com/blogs/itil-service-value-chain/
- https://www.beyond20.com/blog/what-is-the-itil-4-service-value-chain/
- https://www.bmc.com/blogs/itil-management-practices/
- https://www.qrpinternational.ch/en/qrp-news/itil-4-continual-improvement/
- https://www.businesscoachingitalia.com/ciclo-di-deming-cose-le-quattro-fasi-a-quali-procedure-si-attua/
- https://www.bernardello.it/blog/information-technology/framework-itil.htm
- https://www.bitil.com/best-practices/devops
- https://itrevolution.com/the-three-ways-principles-underpinning-devops/

- https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr/65-Fingerpointynessproblem_argggh_xed_freaking_out
- https://www.javatpoint.com/devops-lifecycle
- https://www.edureka.co/blog/devops-lifecycle/
- https://aws.amazon.com/it/microservices/
- https://vitolavecchia.altervista.org/differenza-e-vantaggi-tra-architettura-monolitica-e-architettura-di-microservizi-in-informatica/
- https://it.nttdata.com/insights/blog/5-caratteristiche-governance-progetto
- https://www.ibm.com/docs/en/control-desk/7.6.1?topic=overview-configuration-items
- file:///C:/Users/giuli/Downloads/2020%20State%20of%20DevOps%20Report.pdf
- https://it.wikipedia.org/wiki/Techedge
- https://techedgegroup.com/it/about/scopri-techedge
- https://www.techedgegroup.com/it/about/uffici
- https://www.techedgegroup.com/hubfs/Techedge_Relazione_Finanziaria_2019- 5.pdf?hsLang=it
- https://it.myservername.com/what-is-user-acceptance-testing#:~:text=Il%20test%20di%20accettazione%20dell'utente%20(UAT)%2C%20noto,pu%C3%B2%20essere%20accettato%20o%20meno.&text=Questa%20convalida%20viene%20eseguita%20dagli,familiarit%C3%A0%20con%20i%20requisiti%20aziendali.
- https://www.digital4.biz/marketing/analisi-swot-cos-e-come-farla/
- http://www.pclinkitalia.com/home/wp-content/uploads/2014/02/Descrizione-linee-guida-ITIL.pd
- https://softwarebusiness.it/ams-il-supporto-in-tempo-reale-per-un-sistema-erp-sempre-attivo-ed-efficiente-2/
- https://www.braincomputing.com/devops-come-portare-metodologia-in-azienda/

BIBLIOGRAFIA

[1] Len Bass, Ingo Weber, Liming Zhu – DevOps, A software Architect's Perspective – Addison-Wesley

[2] Jennifer David, Katherine Daniels – Effective DevOps – O'Reilly

[3] Gene Kim, Jeze Humble, Patrick Debois, John Willis – The DevOps Handbook – IT Revolution

RINGRAZIAMENTI

Mi risulta davvero difficile poter esprimere tutta la mia gratitudine nei confronti di tutte le persone che mi hanno sostenuto in questi anni. Innanzitutto, desidero ringraziare tutte le persone che ho avuto di incontrare in Techedge, in particolare: Luigi, Andrea, Enrica e Chiara, grazie per i vostri preziosi insegnamenti e la vostra disponibilità, inutile dire che senza di voi questa tesi non sarebbe potuta esistere.

Il mio pensiero va poi ai miei genitori che con i loro sacrifici non mi hanno mai fatto mancare nulla e mi hanno sempre sostenuto in ogni singolo momento di questo percorso. Ai miei cugini, che con i loro successi e la loro intelligenza mi hanno sempre spronata a dare il meglio. Ai miei zii, che nonostante la distanza conservano sempre una buona parola e un pensiero per la loro nipote preferita.

Ringrazio il Politecnico per avermi permesso di incontrare persone che resteranno indelebili nel mio cuore, in particolare Elisa e Simona con il quale ho condiviso e condividerò ancora tanto. Ringrazio poi le amiche di sempre che sono come una seconda famiglia per me e che hanno creduto nelle mie potenzialità fin dall'inizio di questo percorso: Cecilia, Alice, Carol, Ambra e Chiara.

Un ringraziamento speciale va poi a Gabriele che, seppur abbia condiviso con me solo gli step finali di questo percorso, è stato fondamentale per non mollare ad un passo dal traguardo poiché mi hai dato il sostegno e la giusta spinta per arrivare a questo giorno sopportando la mia ansia e le mie insicurezze.

Infine, un grazie va anche a me stessa per non aver mollato anche quando sembrava che non ci fossero altre alternative, sei forte e devi continuare a dimostrarlo sempre.