CAPCO

REFACTORING DEI SISTEMI LEGACY BANCARI CON IL SUPPORTO DELL'INTELLIGENZA ARTIFICIALE

BUONA LA PRIMA!

L'imperativo per i Chief Information Officer delle banche è evolvere verso i sistemi informativi più efficaci/efficienti e compiere la migrazione verso il cloud.

Il lavoro da fare è notevole in tutte le banche, anche a causa delle numerose fusioni degli anni recenti, che hanno sovrapposto sistemi con linguaggi diversi.

I sistemi informativi di una banca, oggi, sono tipicamente fondati su applicazioni monolitiche, ciascuna delle quali svolge centinaia di funzioni. Dobbiamo immaginarlo, fatte le dovute proporzioni, come l'interruttore generale di un'abitazione che governa i diversi servizi.

Riorganizzare questi monoliti in una serie di microservizi, separabili, che possono migrare in modo indipendente ed essere scalati in modo flessibile ed efficiente ha dimostrato di portare una riduzione dei costi e eliminare gli ostacoli nella migrazione verso il cloud.

I microservizi incrementano l'efficienza delle banche nelle attività di sviluppo, semplificano la distribuzione, rafforzano la resilienza la scalabilità, accelerano la trasformazione digitale.

C'è, tuttavia, da gestire un problema economico perché il costo di questa evoluzione (per la migrazione sul cloud e altri costi operativi) cresce con il numero di microservizi. Qual è il numero di microservizi che garantisce il giusto equilibrio tra prestazioni e costi? Come organizzare al meglio i microservizi per ottenere la massima efficienza? Come costruire la roadmap di modernizzazione tecnologica per eliminare progressivamente i monoliti di oggi interferendo il meno possibile con il sistema informativo nel suo complesso?

L'IA oggi può contribuire a rispondere a queste domande come sta sperimentando sul campo Capco con una tecnologia che può esaminare milioni di righe di codice in pochi minuti facilitando il refactoring.

Considerando le innumerevoli modalità per suddividere le funzioni di un'applicazione legacy, va valutato come riorganizzarle in una struttura di servizi che mantenga gli stessi servizi di business, garantendo al contempo che i sistemi sottostanti rimangano scalabili, sicuri e flessibili anche in futuro, senza incorrere in costi insostenibili per il cloud. Va inoltre considerato che Il CIO ha il controllo dei risultati del refactoring e le decisioni prese all'inizio del processo di modernizzazione dell'applicazione influenzeranno profondamente e a lungo termine il costo totale della soluzione scelta.

Il lavoro di Capco prevede un'analisi iniziale automatizzata del codice delle applicazioni legacy, indipendentemente dal linguaggio di programmazione utilizzato, per valutare la configurazione del sistema. Successivamente, viene creata una tassonomia dei dati e delle funzioni aziendali dei sistemi legacy e delle loro applicazioni monolitiche, basandoci su standard di settore come il Banking Industry Architecture Network (BIAN), ISO 20022 o altri standard in uso all'interno dell'organizzazione.

La metodologia Capco sfrutta l'analisi dei dati supportata dall'apprendimento automatico per determinare il numero ottimale di servizi in cui suddividere i monoliti, garantendo un equilibrio tra qualità e risorse. Il risultato è un design dei servizi ottimale che massimizza l'efficienza. Un approccio che allinea i servizi in base al modo in cui utilizzano e condividono i dati, piuttosto che solo sulla loro funzione e permette di renderli più modulari e scalabili. Viene inoltre fornita una roadmap di refactoring basata sulla nostra metodologia che assegna la priorità alla separazione di ciascun servizio esistente. Questo garantisce un approccio sistematico e standardizzato al refactoring dei monoliti, facilitando anche la futura manutenzione del sistema risultante.

PROBLEMI DELLE APPLICAZIONI MONOLITICHE

Negli ultimi anni, la necessità di scalabilità, agilità e miglioramento della user experience digitale ha spinto le istituzioni finanziarie ad adottare approcci architetturali moderni.

Tra questi vi sono il passaggio ad architetture a microservizi, l'adozione di architetture basate su API, la containerizzazione e le tecnologie native del cloud. Tuttavia, la modernizzazione delle applicazioni monolitiche legacy rappresenta una sfida davvero importante. Si tratta spesso di applicazioni di grandi dimensioni, complesse e critiche dal punto di vista operativo, hanno spesso decenni di vita, con documentazione obsoleta o inesistente. Gli sviluppatori che le hanno progettate e costruite non lavorano più nell'azienda e le competenze necessarie per gestirle sono difficili da trovare sul mercato.

Le difficoltà seguenti sono particolarmente rilevanti per le grandi aziende specialmente nel settore dei servizi finanziari, dove le applicazioni monolitiche legacy costituiscono ancora il "core" degli ecosistemi bancari centrali:

- **Difficoltà nell'introduzione di cambiamenti** – un'applicazione monolitica è tipicamente costruita come un unico sistema in cui le funzioni che la costituiscono sono strettamente e permanentemente interconnesse. Poiché queste applicazioni spesso fanno parte del sistema di "core banking", tale limitazione ha ripercussioni negative su qualsiasi nuova iniziativa di sviluppo di prodotti o rielaborazione di prodotti, specialmente quando sono richieste integrazioni di sistemi complesse.
- **Difficoltà di scalabilità in relazione ai costi** – Le applicazioni legacy stesse sono difficili da scalare in orizzontale in modo efficiente dal punto di vista dei costi. Progettate originariamente per gestire un carico specifico, sono difficili da scalare orizzontalmente per supportare la crescita o gestire le fluttuazioni della domanda. Queste applicazioni risalgono a molto prima dell'avvento della banca digitale e sono poco adattabili alle nuove esigenze di mercato come una pressione ulteriormente aggravata dalla pandemia di COVID-19 e dalle sue ripercussioni. In questo contesto, scalare un'applicazione per soddisfare una domanda crescente richiede la scalabilità dell'intero sistema, risultando così molto costoso.
- **Alto grado di rischio operativo** – Sta diventando sempre più difficile trovare esperti in grado di supportare la tecnologia su cui su cui si basano le applicazioni legacy, spesso sviluppate con linguaggi di programmazione e tecnologie di database superati o addirittura obsoleti. Di conseguenza, il mantenimento e il supporto di queste applicazioni stanno diventando sempre più costosi e complessi.

GRANULARITÀ DEI MICROSERVIZI

Scomporre un'applicazione monolitica in microservizi più piccoli e meno interdipendenti semplifica notevolmente l'identificazione e l'isolamento dei singoli componenti, consentendo di scalarli autonomamente per soddisfare esigenze specifiche. Questo permette alle aziende di allocare le risorse in modo più efficiente. Inoltre, l'uso dei microservizi permette di rilasciare nuovamente singole funzionalità senza compromettere l'intera applicazione principale. Secondo una ricerca di Statista, inoltre, oltre il 50 percento delle aziende che utilizzano i microservizi ritiene che siano molto o estremamente importanti per l'operatività dell'organizzazione. Uno studio indica che il 33 percento dei partecipanti prevede di ristrutturare le proprie applicazioni in microservizi nel periodo 2022-2024, mentre un'altro rileva che fino all'85 percento delle grandi aziende utilizza già i microservizi.

Le principali considerazioni per l'adozione di un approccio basato su microservizi sono le seguenti:

- **Molte modalità per "dividere la torta"** – Un'applicazione monolitica può essere scomposta in una serie di microservizi in molti modi diversi, dai mini-monoliti (l'opzione "bassa granularità") ai nanoservizi ("alta granularità").
- **La scelta è tua** – La granularità del servizio che adotti sarà una scelta consapevole, così come il livello di accoppiamento (interdipendenza) tra i servizi che costituiscono un'applicazione.
- **Implicazioni sui costi e sulle prestazioni** – Le scelte architetturali iniziali avranno un impatto significativo e duraturo sia sui costi operativi del cloud che sull'agilità dell'organizzazione IT che supporta l'applicazione. Maggiore è l'indipendenza dei servizi (ovvero minore è il loro accoppiamento), maggiore sarà l'autonomia e l'agilità dei team che li implementano. Determinare il livello ottimale di granularità influisce anche sull'efficienza del sistema di destinazione, migliorando l'esperienza del cliente e riducendo i costi complessivi.
- **Non c'è via di ritorno** – È necessaria una valutazione molto attenta per determinare il percorso ideale da seguire, poiché una volta presa la decisione su come scomporre il monolite, sarà complesso e costoso tornare indietro o cambiare approccio.

Per ottenere i migliori risultati, è necessario bilanciare gli elementi - la granularità ottimale ed il design ottimale dello stack dei servizi. Esamineremo come raggiungere entrambi.

GRANULARITÀ OTTIMALE DEI MICROSERVIZI: RELAZIONE CON I COSTI ASSOCIATI AL CLOUD

Le decisioni architetturali e di progettazione fondamentali non solo influenzano la qualità e la stabilità di una soluzione basata su microservizi, ma hanno anche un forte impatto sui costi operativi in quanto le tariffe dei fornitori di servizi cloud (CSP) tendono ad essere elevate. Quando si suddivide un'applicazione monolitica in una serie di microservizi, esiste un livello ottimale di granularità che, se applicato in modo coerente, garantirà la minimizzazione del conto mensile del CSP.

La metodologia Capco, basata sull'apprendimento automatico, utilizza principi di data science per trovare il giusto equilibrio.

Ad esempio, consideriamo il costo della migrazione e dell'hosting di un'applicazione monolitica nel cloud. La tabella seguente illustra l'andamento dei costi CSP quando questa applicazione monolitica viene ricostruita come una serie di microservizi.

IMPATTI DEL REFACTORING DI APPLICAZIONI MONOLITICHE SUI COSTI DEL CLOUD

Componenti di costo	**Impatto percentuale sul totale costi CSP**	**Impatti sui costi CSP in uno scenario a microservizi** (si tratta di confrontare i costi CSP della migrazione ad una collezione di microservizi rispetto alla migrazione in cloud di un'unica applicazione monolitica)
Storage		I volumi di dati memorizzati su storage restano gli stessi per cui I costi correlati sono simili
Traffico dati (Ingresso/ Uscita)		I volumi di dati scambiati restano gli stessi per cui I costi correlati sono simili
Overheads		Un aumento lineare dei costi dovuto all'aumento del processo di delivery, gestione e monitoring
Network		Incremento dei costi superlineare dovuto all'aumento di traffico dovuto ai microservizi che comunicano tra loro
Computazionale		Diminuzione logaritmica dei costi dovuta all'aumentata scalabilità dell'ecosistema dei microservizi

E' evidente come l'aumento della granularità dei microservizi influisca sui costi complessivi del cloud. Suddividere un'applicazione monolitica in un numero di servizi pari al numero di funzioni che essa contiene renderà i servizi più indipendenti e più facili da gestire, ma meno efficienti. Questo perché i servizi devono comunque comunicare tra loro per svolgere le loro funzioni, comportando un incremento dei costi.

La figura seguente mostra la relazione tra due forze in competizione: il numero di microservizi e i costi cloud associati, utilizzando come punto di riferimento il costo di hosting su cloud del monolite esistente. La relazione è logaritmica: i costi diminuiscono all'aumentare del numero di microservizi inizialmente, per poi invertire direzione e superare il punto di partenza una volta che il numero di microservizi supera una certa soglia.

Il punto più basso in questa curva di relazione corrisponde al livello ottimale di granularità dei servizi e al costo più basso, identificato come il "numero magico". Concentrandosi su questo numero, la metodologia adottata da Capco garantisce l'individuazione del punto in cui si possono ottenere i maggiori benefici dal design dell'architettura di destinazione con il costo più basso possibile.

IMPATTO DELLA GRANULARITÀ DEI MICROSERVIZI SUI COSTI DEL CLOUD

DESIGN OTTIMALE DELLO STACK DI MICROSERVIZI: UNA SOLUZIONE BASATA SULL'INTERCONNESSIONE DEI DATI

La sfida successiva è definire uno stack di servizi in modo da minimizzare i costi operativi senza perdere le qualità che rendono attraente un'architettura a microservizi.

Qui entrano in gioco due considerazioni chiave:

- **Scalabilità** – più riusciamo a scalare, più basso sarà il costo.
- **Isolamento dei servizi** – più isoliamo i servizi l'uno dall'altro (ovvero minimizziamo la dipendenza intra-servizio o "chatter"), più basso sarà il costo e più alta sarà la performance complessiva dell'applicazione.

Sia la scalabilità che l'isolamento dei microservizi sono strettamente legati al modo in cui questi utilizzano e condividono i dati. I microservizi spesso richiedono livelli di scalabilità diversi a seconda del tasso di operazioni di lettura/scrittura dei dati sottostanti e devono frequentemente comunicare tra loro, poiché un microservizio può possedere i dati necessari affinché un altro possa eseguire una determinata funzionalità.

Pertanto, definire i microservizi insieme a una tassonomia dei dati, che stabilisca una gerarchia di classi di dati basata su caratteristiche comuni, piuttosto che adottare un approccio puramente orientato al dominio o alle capacità aziendali, che può comportare duplicazioni e altre inefficienze, genererà servizi che sono:

- **Intrinsecamente più scalabili** – poiché progettati per gestire gli oggetti di dati utilizzati più frequentemente, in base alle esigenze funzionali del microservizio.
- **Intrinsecamente meglio isolati** – poiché devono consultare altri microservizi meno frequentemente per leggere e scrivere dati.

In definitiva, utilizzando questo approccio orientato ai dati per determinare il livello ottimale di granularità dei servizi si ottengono i seguenti risultati:

- **Performance migliore** – un miglior isolamento, cioè avere servizi meno strettamente accoppiati tra di loro, genera un miglioramento delle prestazioni dell'applicazione.
- **Costi inferiori** – la metodologia Capco suggerisce il minor numero di servizi che forniscono sia scalabilità che modularità (isolamento), corrispondente quindi alla spesa più bassa possibile per CSP.

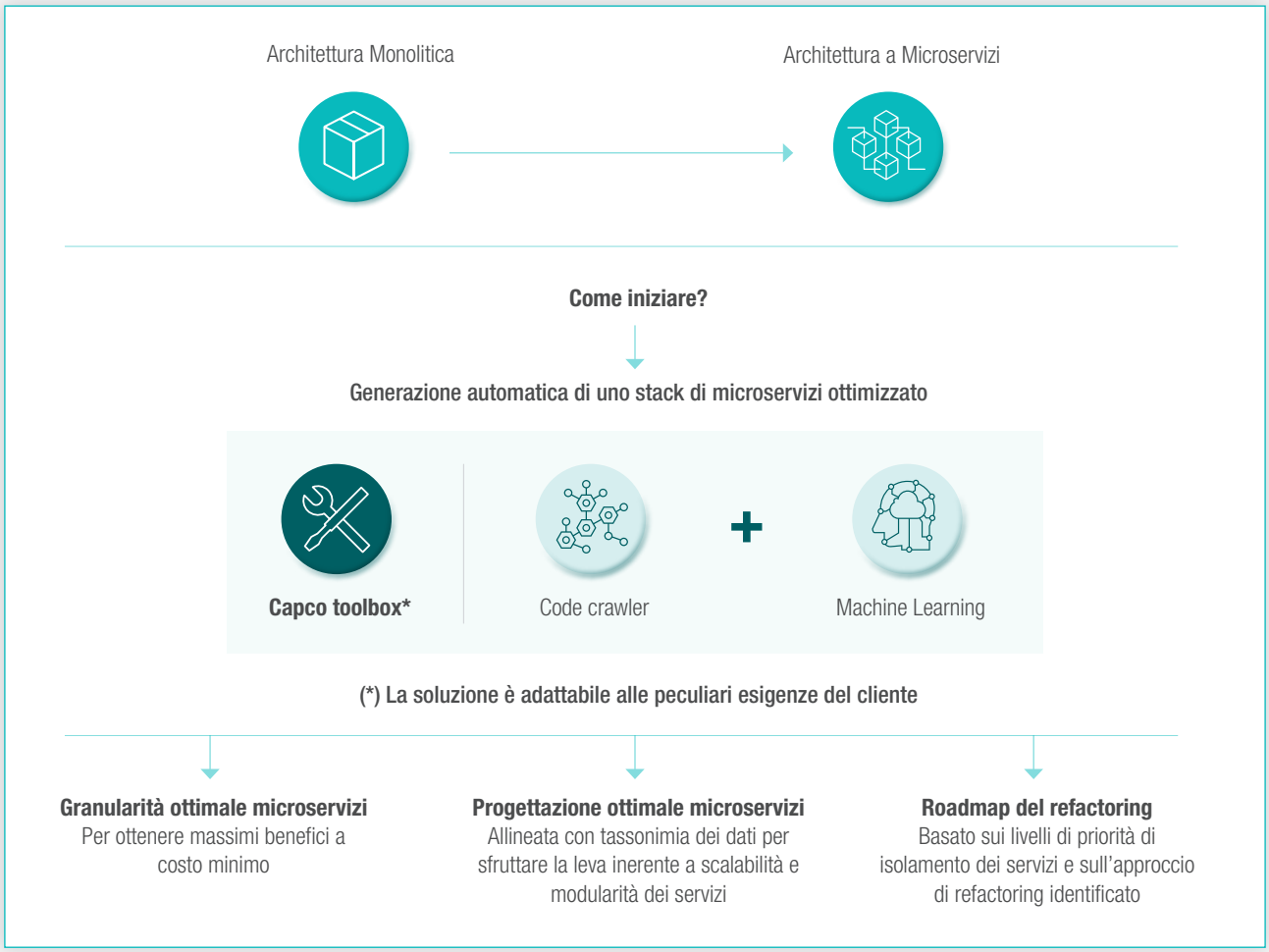
METODOLOGIA CAPCO: OVERVIEW

La nostra soluzione è personalizzata per lo scenario specifico di ciascun cliente, tenendo conto di eventuali vincoli esistenti, come contratti con fornitori di piattaforme specifici. Il nostro crawler di codice, supportato dall'intelligenza artificiale, scansiona automaticamente il codice sorgente legacy (scritto in COBOL, Java o altri linguaggi di programmazione) per creare un profilo completo di tutte le funzionalità e degli oggetti di dati che queste manipolano. Estrae anche eventuali commenti incorporati nel codice per creare documentazione in tempo reale.

Un algoritmo di machine learning raggruppa poi le capacità/funzionalità in base ai pattern dei dati e fornisce una raccomandazione per il numero di microservizi in cui suddividere il monolite. Identifica inoltre i dati che ciascun microservizio dovrebbe gestire, ottimizzando l'isolamento del microservizio, ed evidenzia quali microservizi causeranno meno interruzioni all'ecosistema se "estratti" dal monolite in anticipo. Questo processo di assegnazione delle priorità imposta la roadmap di migrazione verso la nuova architettura precedentemente menzionata.

L'automazione del refactoring dell'architettura in obsolescenza offre i seguenti significativi vantaggi:

REFACTORING AUTOMATIZZATO DELL' ARCHITETTURA LEGACY



- **Velocità:** adattando la metodologia alla configurazione specifica del cliente e utilizzando l'intelligenza artificiale, la fase di discovery del codice legacy, estremamente importante, viene ridotta da mesi a giorni.
- **Qualità:** Un approccio standardizzato e orientato ai dati, rispetto a uno focalizzato sul dominio funzionale, per il refactoring e la migrazione dei sistemi legacy produce risultati coerenti che permettono di stimare con precisione i costi. Questo aiuta a definire e a dare priorità alla roadmap del refactoring e a convalidare le strategie di refactoring.
- **Costi ridotti:** lo stack di servizi è ottimizzato per minimizzare il traffico tra i servizi, riducendo così i costi di rete del cloud. Evitando la creazione di microservizi ultra-granulari, si riducono i costi di monitoraggio e di overhead. Inoltre, i microservizi risultanti sono intrinsecamente più scalabili, riducendo i costi computazionali associati. La fase accelerata di discovery del codice richiede meno esperti coinvolti, eliminando così i costi di formazione. La nostra metodologia utilizza diversi approcci di clustering dei dati per arrivare alla soluzione più completa.

CASO DI SUCCESSO: REFACTORING DI UN APPLICATIVO PER UNA BANCA TIER 1

Come parte di un programma di migrazione su larga scala verso il cloud a livello aziendale, il nostro cliente - una banca di primo livello - stava cercando acceleratori e tecniche per dare la priorità alla migrazione delle applicazioni legacy verso il cloud e ottimizzare il complesso processo di trasformazione. Il cliente ha utilizzato con successo la nostra soluzione per abbreviare il time-to-market e accelerare il proprio percorso di modernizzazione dei sistemi, in particolar modo nella fase di pianificazione.

Utilizzando la nostra metodologia, abbiamo sviluppato uno strumento per semplificare e accelerare l'analisi statica del codice in obsolescenza, senza impattare sull'ambiente di produzione. Successivamente, gli output generati da questo strumento sono stati utilizzati per alimentare algoritmi di apprendimento automatico, aiutando a definire la forma e la struttura di una raccolta ottimizzata di microservizi.

Obiettivi del cliente:

- Evolvere da un'architettura di sistema legacy ad un'architettura scalabile in modo ottimizzato per garantire la qualità dei risultati ottenuti, con costi ridotti e bassa complessità nella futura manutenzione.
- Sviluppare questa evoluzione architetturale verso i microservizi in modo organizzato e standardizzato, seguendo i principi di governance, agilità e automazione e facendo leva sull'apprendimento automatico basato su intelligenza artificiale.
- Assicurare il delivery con massime prestazioni e stabilità dell'applicazione, selezionando attentamente la granularità del livello dei servizi per evitare errori difficili ed costosi da correggere.

Cosa abbiamo consegnato:

- Automazione: Esecuzione dell'analisi automatizzata del codice sorgente, applicando e utilizzando l'elaborazione a linguaggio naturale dove applicabile.
- Definizione della tassonomia: Sviluppo di una tassonomia dati e business utilizzata dal sistema legacy. Questa tassonomia può basarsi su standard di mercato come BIAN o ISO200022 o altro già in uso dall'organizzazione.
- Strategia e roadmap: Identificazione della granularità ottimale dei microservizi e creazione di una roadmap di migrazione in linea alla governance del progetto del cliente.

Nel dettaglio, durante un periodo di 18 mesi e con un team di 10 esperti in materie tecnologiche, abbiamo raggiunto i seguenti risultati:

- Integrato il nostro strumento con il repository del codice COBOL per recuperare e leggere automaticamente il codice sorgente in obsolescenza.
- Sviluppato un crawler di codice per programmi e job COBOL:

- Progettato un parser di componenti COBOL che ci ha permesso di costruire una visualizzazione-XY delle funzionalità e degli oggetti di dati che ogni funzionalità manipola.
- La visualizzazione XY è diventata un input per il passaggio successivo del processo - il refactoring tramite diverse tecniche di apprendimento automatico non supervisionato.
- Selezionato il miglior metodo di refactoring per ciascun esercizio di refactoring.

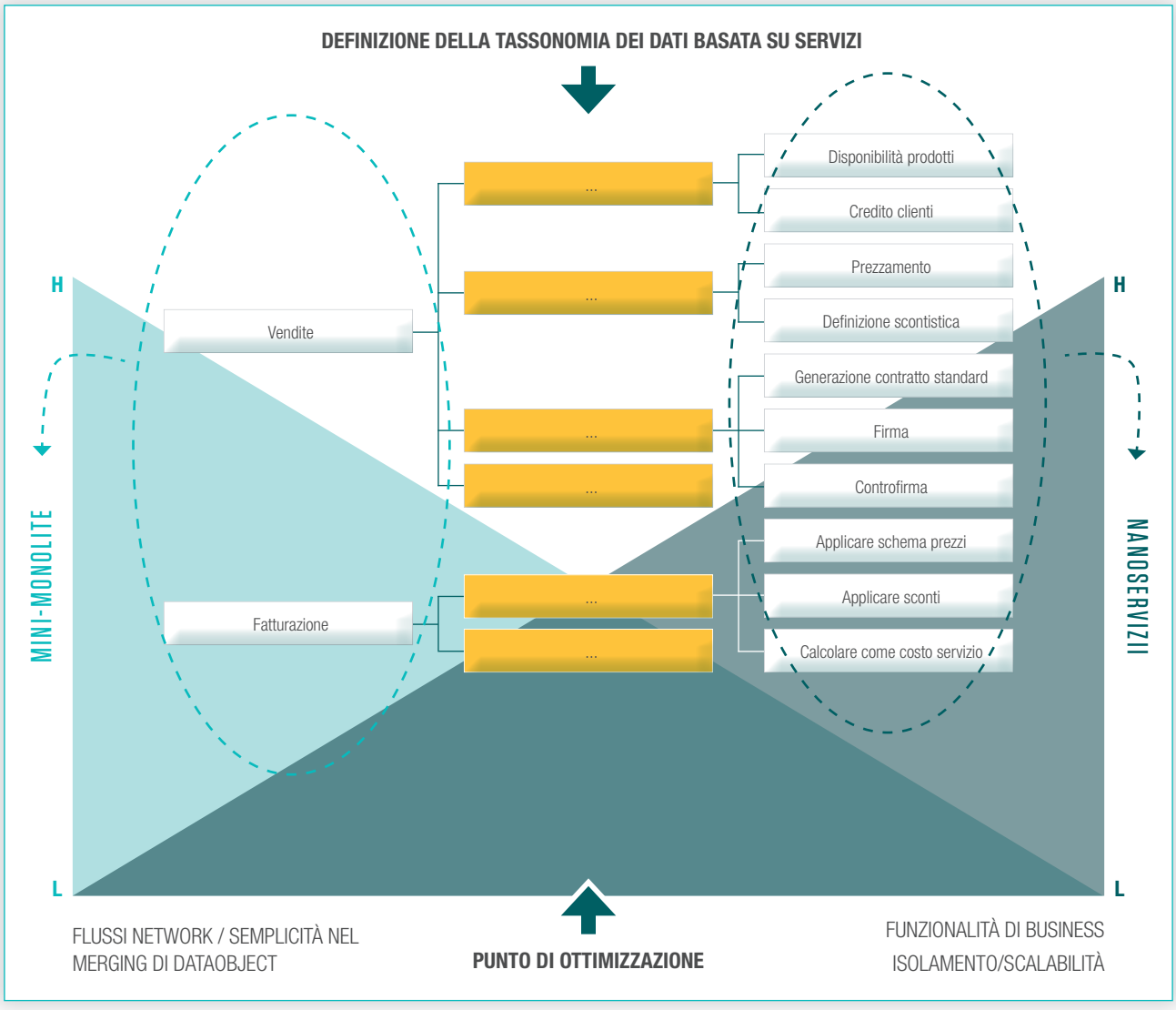
- Creato un report che ha aiutato gli stakeholder ed i decision maker a comprendere i risultati e le conseguenze dell'output della nostra analisi e delle raccomandazioni di refactoring.

Benefici per il cliente

Utilizzando il nostro strumento automatizzato, la banca è riuscita a refattorizzare con successo le sue applicazioni monolitiche basate su COBOL in microservizi candidati per oltre 20 aree di business. Questo ha inoltre permesso di creare un approccio standard riutilizzabile per definire tassonomie funzionali e tecniche da applicare in future iniziative di refactoring dei monoliti e migrazioni dei sistemi. Gli ulteriori impatti includono:

- La fase di pianificazione del progetto è stata ridotta da sei mesi a tre settimane, con meno esperti richiesti, eliminando quindi i costi di formazione.
- Sono stati elaborati circa 6.000 programmi legacy COBOL, contenenti complessivamente oltre 1,5 milioni di righe di codice (l'applicazione più grande consisteva in oltre 200.000 righe di codice).
- Etichettate e mappate circa 40.000 tabelle del database mainframe utilizzando l'etichettatura delle funzionalità per facilitare la migrazione verso il cloud.

MASSIMIZZAZIONE DELL'ISOLAMENTO DEI MICROSERVIZI MINIMIZZANDO CONTEMPORANEAMENTE LA COMUNICAZIONE TRA MICROSERVIZI



CONCLUSIONI E PROSSIMI PASSI

Il refactoring e la modernizzazione dei sistemi bancari legacy sono processi importanti ed è vitale svolgerli bene. E' quindi fondamentale stabilire la granularità e la struttura ideale dei microservizi che permetterà di progettare e implementare un'architettura che offrirà tutta l'agilità, la scalabilità e la sicurezza richieste da una banca moderna.

Un design architetturale inefficiente e l'impiego di più microservizi del necessario possono rapidamente tradursi in bollette mensili esorbitanti per i servizi cloud, come rete, computazione, monitoraggio e altri servizi. Sebbene il cloud offra molte possibilità, comporta anche dei costi. Prendere le decisioni architetturali corrette all'inizio del processo di progettazione permetterà ai CIO di avere un maggiore controllo sui costi operativi del sistema nel lungo termine.

Un design ottimizzato dei microservizi influenzerà significativamente le prestazioni, l'integrità e la qualità dei dati, dei prodotti e dei servizi del sistema bancario di destinazione. È importante non solo eseguire questo processo correttamente fin dall'inizio, ma anche procedere in modo graduale. La metodologia Capco aiuta a definire il percorso ottimizzato per un refactoring completo, assegnando priorità ai microservizi meno interconnessi con gli altri, riducendo così le interruzioni durante la loro rimozione.

Infine, il refactoring di un monolite comporta la necessità che i nuovi componenti del sistema (microservizi) comunichino tra loro, generando un overhead di rete che non esiste in un'architettura monolitica. Creare microservizi troppo dipendenti l'uno dall'altro, o che necessitano di scambiare troppe informazioni o farlo più frequentemente del necessario (diventando così troppo "chiacchieroni"), rende l'architettura a microservizi inefficiente e costosa, compromettendone i benefici. La metodologia di Capco affronta specificamente questo problema, allineando i microservizi con una tassonomia dei dati per garantire che siano il più isolati e scalabili possibile.

In sintesi, considerati i considerevoli costi del cloud e le implicazioni che le scelte architetturali e di progettazione hanno sulle prestazioni delle applicazioni, raccomandiamo ai CIO di adottare un approccio automatizzato e orientato ai dati, piuttosto che focalizzato sui domini funzionali, per trovare la soluzione ottimale per il refactoring delle applicazioni monolitiche dei sistemi legacy.

La metodologia Capco, applicata fin dalle prime fasi del processo di progettazione, fornirà i dati e la fiducia necessari per prendere decisioni informate, garantendo il successo a lungo termine sia del refactoring dei sistemi in obsolescenza sia delle iniziative di sviluppo da zero.

AUTORI

Gerhardt Scriven, Managing Principal
Marcel Braga, Principal Consultant

CONTATTI

Paolo La Torre, Partner, paolo.latorre@capco.com
Massimiliano Orso, Principal Consultant, massimiliano.orso@capco.com

CAPCO

Capco è una società di Wipro che opera a livello globale nella consulenza tecnologica e direzionale specializzata nel guidare la trasformazione dei settori dell'energia e dei servizi finanziari. Capco agisce all'intersezione tra business e tecnologia combinando pensiero innovativo con una conoscenza del settore senza pari per accelerare le iniziative digitali nei settori del banking e dei pagamenti, dei mercati dei capitali, della gestione patrimoniale e degli asset, delle assicurazioni e dell'energia. La capacità innovativa di Capco prende vita attraverso la sua cultura premiata Be Yourself At Work e attraverso il suo talento diversificato.

Per saperne di più, visita www.capco.com o seguici su LinkedIn, Instagram, Facebook e YouTube.

UFFICI NEL MONDO

APAC
Bengaluru – Electronic City
Bengaluru – Sarjapur Road
Bangkok
Chennai
Gurugram
Hong Kong
Hyderabad
Kuala Lumpur
Mumbai
Pune
Singapore

MEDIO ORIENTE
Dubai

EUROPA
Berlino
Bratislava
Bruxelles
Dusseldorf
Edimburgo
Francoforte
Ginevra
Glasgow
Londra
Milano
Parigi
Vienna
Varsavia
Zurigo

NORD AMERICA
Charlotte
Chicago
Dallas
Hartford
Houston
New York
Orlando
Toronto

SUD AMERICA
São Paulo

WWW.CAPCO.COM

CAPCO
a wipro company