ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Dipartimento di Informatica — Scienza e Ingegneria
Corso di Laurea Triennale in Informatica per il Management

Digitalizzazione dei flussi finanziari aziendali: un approccio API-centric all'integrazione bancaria in sistemi ERP proprietari

Relatore:
Prof.
Davide Sangiorgi

Presentata da:
Linda Gotti

Sessione Luglio 2026
Anno Accademico 2025/2026

Abstract

Ogni giorno, migliaia di aziende italiane gestiscono i propri pagamenti aprendo un portale bancario, scaricando un file, caricandolo manualmente e aspettando.
Un processo che funziona, ma che appartiene a un'epoca in cui i sistemi non sapevano parlarsi.

Questa tesi racconta come si cambia un processo del genere dall'interno — senza toccare il cuore di un software che funziona, senza disorientare chi lo usa da anni, e senza aspettare che qualcuno riscriva tutto da zero.

Il contesto è **SAM ERP 2.0** di Centro Software S.p.A., un gestionale proprietario diffuso tra le PMI italiane.
L'obiettivo era integrarlo direttamente con il sistema bancario, trasformando un flusso manuale e frammentato in un processo automatizzato, tracciabile e governabile in ogni sua fase — dall'invio delle disposizioni di pagamento fino alla riconciliazione degli esiti.

La sfida non era tecnica. Era architetturale: intervenire su un sistema **closed-source**, senza accesso al codice sorgente, introducendo un cambiamento profondo che gli utenti non dovevano nemmeno percepire. La soluzione ha preso la forma di un meccanismo di intercettazione non invasiva degli eventi applicativi, di un modello dati gerarchico capace di rappresentare l'intero ciclo di vita di una transazione finanziaria, e di un'integrazione **API-centric** con il gateway bancario.

Il risultato è un sistema in cui i pagamenti si aggregano, si validano, si trasmettono e si riconciliano — in modo automatico, sicuro e verificabile, in cui ogni stato intermedio è visibile, interrogabile e governabile senza dipendere dalla trasparenza della controparte bancaria.

Quello che questa tesi dimostra, alla fine, è che digitalizzare un processo aziendale non significa riscriverlo. Significa capirlo abbastanza in profondità da poterlo cambiare senza romperlo.

Indice

Capitolo 1

Stato dell'arte e analisi dei requisiti

1.1 Il ruolo dei sistemi ERP nelle PMI italiane

Negli ultimi decenni, i sistemi ERP (Enterprise Resource Planning) hanno assunto un ruolo centrale nell'infrastruttura informativa delle imprese, supportando la gestione integrata dei processi operativi, amministrativi e finanziari.
In particolare, nel contesto delle piccole e medie imprese italiane (PMI), tali sistemi rappresentano il fulcro delle attività aziendali, consentendo la gestione coordinata di contabilità, fatturazione, magazzino e flussi di pagamento.

Nel panorama italiano, caratterizzato da una forte predominanza di PMI, l'adozione dei sistemi ERP presenta peculiarità distintive rispetto ai modelli tipici delle grandi organizzazioni. Come evidenziato in letteratura [3], l'introduzione di tali sistemi ha rappresentato uno dei fenomeni più rilevanti nel campo dell'Information Technology per questo segmento, influenzando in modo significativo i processi decisionali e operativi delle imprese.
Le PMI tendono infatti a privilegiare soluzioni sviluppate da fornitori locali, in grado di offrire prodotti meno complessi, più economici e maggiormente adattabili alle specifiche esigenze operative.
Questi sistemi, comunemente definiti ***light ERP***, costituiscono una risposta pragmatica alla necessità di digitalizzazione, permettendo alle organizzazioni di ottenere benefici immediati senza affrontare gli elevati costi e la complessità delle piattaforme enterprise tradizionali.

Tuttavia, tale approccio introduce un compromesso tra semplicità e capacità evolutiva. In particolare, si osserva il cosiddetto "paradosso dell'ERP": sistemi inizialmente flessibili in fase di configurazione tendono a diventare progressivamente rigidi una volta messi a regime. Questa rigidità operativa rappresenta un limite significativo nei contesti in cui i requisiti applicativi evolvono rapidamente, come nel caso dell'integrazione con servizi esterni e piattaforme finanziarie.

1.2 Case Study: Centro Software S.p.A. e integrazione SAM ERP 2.0

Il presente lavoro si inserisce nel contesto di **Centro Software S.p.A.**, azienda italiana attiva nello sviluppo di soluzioni gestionali per piccole e medie imprese. Il sistema oggetto di analisi è **SAM ERP 2.0**, piattaforma proprietaria sviluppata in ambiente Delphi e destinata alla gestione integrata dei processi aziendali, inclusi contabilità, fatturazione e flussi finanziari.

Il gestionale è distribuito secondo un modello **closed-source**, in cui il codice sorgente non è accessibile agli utilizzatori né agli sviluppatori esterni.
Le personalizzazioni e le estensioni funzionali sono quindi realizzate esclusivamente attraverso i meccanismi di estensione messi a disposizione dal framework applicativo, come eventi, componenti standard e interfacce predefinite.

Nel contesto del tirocinio, lo sviluppo è avvenuto all'interno di un ambiente di test configurato su SAM ERP 2.0, con accesso completo al gestionale e a un'istanza dedicata del database SQL Server associata a un'azienda fittizia.
L'attività progettuale si è tuttavia svolta senza accesso al codice sorgente e senza possibilità di modifica del core applicativo, che è stato trattato come componente non modificabile.
L'integrazione è stata quindi progettata secondo un paradigma di interazione esterna al sistema, basato sull'utilizzo dei meccanismi di estensione disponibili e su componenti indipendenti e non intrusivi.

Durante il periodo di tirocinio, SAM ERP 2.0 è stato soggetto a evoluzioni progressive, con un ciclo di rilascio medio di circa quattro mesi. In particolare, il sistema ha attraversato la transizione dalla versione 6.10 alla 6.12, evidenziando la necessità di garantire compatibilità tra versioni differenti. Parallelamente, sono ancora presenti installazioni in ambiente produttivo basate su release della serie 5.x, rendendo fondamentale la retrocompatibilità delle estensioni sviluppate.

1.2.1 Perimetro del tirocinio e contributo della tesi

Il lavoro complessivo descritto in questa tesi si articola in due fasi distinte, sebbene strettamente connesse, che è opportuno distinguere con precisione.

La **prima fase**, svolta nell'ambito del tirocinio curriculare (dicembre 2025 – marzo 2026), ha posto le fondamenta architetturali su cui si costruisce tutto il lavoro successivo. Ha riguardato la creazione e la gestione degli oggetti complessi — *PaymentOrders*, *BankReceipts* e *ManualPaymentOrders* — e la loro completa integrazione nella logica del gestionale, attraverso il meccanismo di intercettazione tramite **Redirect Event**, la logica di associazione massiva dei titoli contabili e l'interfaccia utente integrata nell'ecosistema di SAM ERP 2.0. I dettagli progettuali e implementativi di questa fase sono approfonditi nel Capitolo 2.

La **seconda fase**, oggetto specifico di approfondimento nella presente tesi, ha riguardato l'integrazione con il gateway bancario e la gestione dei flussi di ritorno. Questa parte del lavoro è stata sviluppata nel periodo successivo al tirocinio, nell'ambito di un contratto di collaborazione con Centro Software S.p.A., e ha richiesto un significativo lavoro di ricerca e sperimentazione autonoma: lo studio delle specifiche API, la progettazione del protocollo di autenticazione e del modello di licensing, l'analisi delle strategie di riconciliazione degli esiti e la definizione del ciclo di vita delle disposizioni.
Trattandosi di specifiche in parte ancora in evoluzione al momento della stesura — la release 6.13 è prevista per settembre 2026 — questa fase ha richiesto la capacità di progettare componenti flessibili in condizioni di incertezza, una competenza che va oltre il contesto formativo del tirocinio e che costituisce il contributo originale più rilevante di questo lavoro.

È importante sottolineare che entrambe le fasi concorrono alla realizzazione di un unico prodotto completo e coerente.
La distinzione tra le due fasi non riguarda quindi la natura del lavoro — che è sempre stato orientato a un obiettivo produttivo reale — ma il grado di autonomia e il tipo di contributo apportato in ciascuna di esse.

1.3 Requisiti funzionali del sistema

I requisiti funzionali del sistema non sono stati definiti in maniera statica a priori, ma sono emersi progressivamente nel corso dello sviluppo attraverso un processo iterativo di analisi, confronto con il team aziendale e recepimento continuo delle esigenze operative e normative del dominio finanziario. Tale modalità riflette un approccio tipico delle metodologie Agile, in cui i requisiti evolvono in funzione del contesto applicativo e del feedback degli stakeholder, consentendo di adattare le funzionalità alle reali necessità operative man mano che queste emergono.

Al termine del ciclo di sviluppo è possibile sintetizzare i requisiti funzionali in un insieme di macro-funzionalità. Per ciascuna viene indicato il comportamento atteso; le modalità implementative e i dettagli architetturali sono approfonditi nel Capitolo 2.

- **RF1 — Gestione delle entità finanziarie**
  Il sistema deve consentire la creazione, modifica, consultazione e persistenza delle entità introdotte dall'estensione: Disposizioni di Pagamento (*Payment Orders*), Ricevute Bancarie (*Bank Receipts*) e Disposizioni di Pagamento Manuali (*Manual Payment Orders*). Tali entità costituiscono il nucleo del modello dati introdotto dal plugin e il punto di riferimento per tutte le operazioni successive.

- **RF2 — Supporto operativo tramite interfaccia utente**
  Il sistema deve fornire interfacce dedicate per le operazioni descritto dall'RF1, mantenendo coerenza visiva e funzionale con le convenzioni operative di SAM ERP 2.0. L'obiettivo è minimizzare l'impatto sugli operatori, evitando la necessità di formazione specifica per l'utilizzo delle nuove funzionalità.

- **RF3 — Aggregazione e gestione massiva delle pendenze**
  L'utente deve poter associare e disassociare in modalità massiva documenti contabili attivi e passivi a specifiche disposizioni o distinte. Questo requisito rispon-

de alla necessità di superare la frammentazione dei dati tipica del sistema standard, consentendo di costruire aggregati coerenti destinati alla generazione e alla trasmissione bancaria.

- **RF4 — Contestualizzazione e filtraggio dei dati**
  Il sistema deve consentire la visualizzazione selettiva delle pendenze associate a una specifica disposizione o distinta, permettendo all'operatore di lavorare all'interno di un contesto operativo delimitato. Il filtraggio deve tenere conto della tipologia di flusso attiva, escludendo le distinte già trasmesse al gateway bancario.

- **RF5 — Validazione dei dati e delle operazioni**
  Prima dell'esecuzione delle operazioni critiche, il sistema deve verificare la correttezza e la consistenza dei dati, impedendo la generazione o la trasmissione di flussi non conformi.
  Questo requisito è particolarmente rilevante nella fase di generazione del file di invio, dove un aggregato ibrido — contenente titoli afferenti a disposizioni differenti — produrrebbe un file rifiutato dal gateway bancario.

- **RF6 — Gestione del ciclo di vita delle disposizioni**
  Le entità finanziarie devono essere soggette a un ciclo di vita controllato, caratterizzato da stati operativi distinti (*Provvisorio*, *Definitivo*, *Inviato*) che ne regolano la modificabilità e l'avanzamento nel processo di elaborazione. Il passaggio di stato deve avvenire in modo automatico al verificarsi degli eventi applicativi corrispondenti.

- **RF7 — Generazione dei flussi bancari**
  Il sistema deve produrre flussi telematici conformi agli standard interbancari adottati dal gateway esterno, utilizzando i dati aggregati nelle disposizioni e nelle distinte.
  I formati supportati comprendono SEPA Credit Transfer, Foreign Transfer, SDD e RiBa, ciascuno soggetto a regole di generazione specifiche, secondo la normativa legale e le regole imposte dal sistema bancario con cui ci interfacceremo.

- **RF8 — Trasmissione automatizzata verso il gateway bancario**
  Il sistema deve consentire l'invio automatico dei flussi generati verso il servizio bancario mediante integrazione API, eliminando la necessità di operazioni manuali di esportazione e caricamento su piattaforme di home banking. Questo requisito rappresenta l'obiettivo principale dell'intera integrazione.

- **RF9 — Monitoraggio e riconciliazione**
  Il sistema deve acquisire e gestire gli esiti restituiti dal gateway bancario, aggiornando l'OperationResult delle disposizioni e garantendo il costante allineamento tra il gestionale e il sistema esterno.
  In caso di errore, l'operatore deve ricevere un messaggio informativo che consenta di individuare e correggere la causa senza perdita di dati.

La classificazione presentata costituisce la base per la progettazione architetturale descritta nei capitoli successivi, in cui per ciascun requisito vengono illustrate le scelte implementative adottate e i meccanismi tecnici che ne garantiscono il soddisfacimento.

1.4 Requisiti non funzionali e vincoli progettuali

Operando in un dominio caratterizzato dalla gestione di dati finanziari e dall'integrazione con servizi bancari esterni, il progetto è stato guidato da una serie di requisiti non funzionali e vincoli architetturali che hanno influenzato in modo significativo le scelte progettuali adottate.

- **RNF1 - Vincolo Black-Box**
  L'estensione deve operare senza modificare il codice sorgente di SAM ERP 2.0, utilizzando esclusivamente i meccanismi di estensione resi disponibili dal framework applicativo.

- **RNF2 - Manutenibilità e separazione delle responsabilità**
  L'architettura deve favorire la modularità e la separazione delle responsabilità, isolando la logica applicativa introdotta dall'estensione in componenti indipendenti e facilmente manutenibili.

- **RNF3 - Compatibilità e retrocompatibilità**
  L'estensione deve mantenere la compatibilità con le differenti versioni del gestionale presenti presso i clienti, garantendo il corretto funzionamento sia sulle release più recenti sia sulle installazioni ancora basate su versioni precedenti.

- **RNF4 - Sicurezza dell'accesso ai servizi esterni**
  L'accesso ai servizi bancari deve avvenire attraverso meccanismi di autenticazione sicuri, limitando l'esposizione delle credenziali e rispettando il principio del *Least Privilege*.

- **RNF5 - Integrità e tracciabilità del dato**
  Il sistema deve garantire la conservazione dell'integrità delle informazioni durante tutte le fasi del processo e consentire la ricostruzione delle operazioni effettuate dagli utenti.

- **RNF6 - Affidabilità e robustezza operativa**
  Il sistema deve gestire correttamente condizioni di errore, malfunzionamenti di rete e indisponibilità temporanee dei servizi esterni, evitando situazioni di inconsistenza tra i sistemi coinvolti.

- **RNF7 - Auditabilità**
  Ogni operazione significativa deve essere registrata attraverso meccanismi di logging persistente, consentendo attività di controllo, verifica e riconciliazione.

- **RNF8 - Usabilità e integrazione con l'ERP**
  Le interfacce introdotte dall'estensione devono mantenere coerenza grafica e funzionale con l'ambiente SAM ERP 2.0.

- **RNF9 - Efficienza operativa**
  L'automazione dei flussi deve ridurre il numero di operazioni manuali richieste agli operatori, diminuendo il rischio di errore umano e migliorando l'efficienza complessiva del processo.

1.4.1 Sintesi del problema

Alla luce del contesto descritto, possiamo ora riassumere così l'obiettivo del presente lavoro: vogliamo progettare un meccanismo di estensione per un sistema ERP proprietario operante in modalità black-box, che consenta l'automazione dei flussi di pagamento e l'integrazione con servizi esterni, garantendo al contempo elevati standard di sicurezza, tracciabilità e manutenibilità.

La risoluzione di tale problematica richiede l'adozione di strategie di integrazione non invasive e l'impiego di architetture event-driven e API-centric, che saranno oggetto di approfondimento nei capitoli successivi.

Capitolo 2

Architettura e Strategie di Estensione

2.1 Il vincolo black-box come driver architetturale

Nel contesto dei sistemi ERP proprietari, la disponibilità o meno del codice sorgente determina quali saranno le strategie di evoluzione del software. Il sistema SAM ERP 2.0 si configura come una piattaforma *closed-source*, imponendo un approccio di tipo *black-box*: l'entità è osservabile esclusivamente attraverso input e output, mentre l'implementazione interna rimane inaccessibile.

Questo paradigma si discosta nettamente dai framework *white-box*, basati sull'estensione per ereditarietà.
Come evidenziato dalla letteratura classica, l'ereditarietà crea un legame statico per cui ogni modifica alla classe base si propaga alle sottoclassi; al contrario, la composizione strutturata tramite interfacce pubbliche mantiene le implementazioni reciprocamente opache, riducendo l'accoppiamento [4].
Sebbene il vincolo *closed-source* precluda il refactoring del nucleo applicativo, esso agisce come driver verso una maggiore maturità progettuale.

Operare all'esterno del codice sorgente riduce drasticamente il *coupling* tra le personalizzazioni e il core dell'ERP, garantendo la preservazione del sistema durante gli aggiornamenti. Touil osserva come, nei modelli a estensione, le componenti che comunicano con l'applicazione base esclusivamente tramite interfacce ad eventi presentino un accoppiamento strutturalmente basso.
Questo riduce la probabilità che le modifiche al core si propaghino come *breaking change*, eliminando l'onere di *merge* manuale tipico dei sistemi a modifica diretta degli oggetti [5].
In quest'ottica, quello che a primo impatto poteva sembrare un grande limite tecnologico si traduce in vantaggio metodologico: l'estensione avviene per intercettazione dei flussi e non per alterazione, forzando una separazione delle responsabilità che protegge l'integrità del sistema legacy.

L'adozione di queste strategie risponde a precisi obiettivi strategici nel ciclo di vita del software [5]:

- **Upgrade-ability:** l'isolamento delle personalizzazioni garantisce la persistenza delle funzionalità aggiuntive a seguito di aggiornamenti, azzerando i costi di migrazione. Tale necessità è coerente con l'approccio di Parr e Shanks, secondo cui la manutenzione di un ERP è una sfida continuativa e non un evento isolato (Parr & Shanks, 2000, come citato in [5]).

- **Contenimento del debito tecnico:** impedendo la commistione tra logiche di business specifiche e codice standard, si evita l'erosione strutturale del sistema, mantenendo una base di codice coerente e sostenibile nel tempo.

- **Modularity e Manutenibilità:** l'incapsulamento delle estensioni in moduli esterni consente interventi localizzati, semplificando le attività di debugging e testing [4].

- **Compliance al principio Open/Closed:** Meyer stabilisce che un modulo debba essere aperto all'estensione del comportamento, ma chiuso alla modifica del proprio codice [6]. L'architettura di SAM ERP 2.0 concretizza questo principio in modo letterale: il *closed* è una condizione imposta dalla natura proprietaria della piattaforma, mentre l'*open* si realizza mediante i meccanismi di estensione esposti dal framework.

L'estensione evolve così da una pratica di *patching* a un processo di integrazione orchestrata, in cui le nuove funzionalità sono introdotte come componenti indipendenti. Questo passaggio, documentato come *best practice* nei contesti industriali con sistemi ERP certificati o *legacy*, rappresenta un riallineamento strategico verso la manutenibilità su larga scala [5]. L'approccio *black-box*, adottato forzatamente per SAM ERP 2.0, si rivela in definitiva una scelta lungimirante e coerente con la filosofia *upgrade-safe* dei framework ERP contemporanei.

2.2 Architettura del plugin di estensione

L'architettura del modulo di estensione è il risultato di un processo di sviluppo iterativo e adattivo, il cui principale scopo è subito stato ben definito ed inquadrato: garantire l'automazione dei processi di pagamento e incasso, introducendo meccanismi di aggregazione, validazione e trasmissione dei flussi bancari.
Questa introduzione vuole ridurre il volume delle attività manuali a carico dell'operatore. Tali attività, infatti, espongono a un maggior rischio di imprecisioni ed errori, causando perdite di tempo in operazioni standard e ripetitive, prive di valore strategico per l'azienda.

Dal punto di vista logico, il plugin è articolato su tre livelli principali:

- **Persistence Layer**, è il motore di memorizzazione del plugin. Si occupa di tradurre i dati logici in record fisici, salvandoli su un database o sul repository aziendale, in modo da mantenere lo stato delle nuove entità in modo affidabile;

- **Presentation Layer**, rappresenta l'interfaccia utente (UI) del plugin ed è quindi il suo volto visibile all'operatore.
  Attraverso maschere, bottoni, griglie, permette all'utente di interagire con i dati

senza uscire dall'ambiente nativo del gestionale.
Questo livello è stato studiato per fornire un'esperienza utente uniforme, integrandosi perfettamente con il look and feel del sistema esistente;

- **Integration Layer**, il vero collante del sistema, paragonabile a un middleware interno. La sua funzione è quella di monitorare e gestire il flusso delle informazioni intercettando gli eventi generati dall'ERP e attivando le logiche del plugin di conseguenza.

Definizione e struttura degli oggetti complessi

La prima fase progettuale ha riguardato la definizione e la validazione delle entità logiche necessarie a supportare il nuovo processo di aggregazione.
Tali entità non rappresentano semplici strutture dati di supporto, ma fungono da punti di snodo strutturali, in quanto traducono dati contabili disomogenei e dispersi in informazioni di cassa standardizzate.

Le entità introdotte sono le seguenti:

- **PaymentOrders**: rappresentano le disposizioni di pagamento e fungono da contenitore logico per l'aggregazione dei mandati di pagamento destinati ai fornitori;
- **BankReceipts**: rappresentano le distinte di incasso e consentono la gestione aggregata delle ricevute bancarie derivanti dal ciclo attivo aziendale;
- **ManualPaymentOrders**: consentono la gestione di pagamenti non derivanti direttamente dai processi contabili standard dell'ERP, come anticipi, rimborsi o altre operazioni straordinarie.
  Tali disposizioni vengono (sin dal momento della creazione) aggregate all'interno di una *PaymentOrder*, in modo da partecipare al medesimo processo di generazione e trasmissione del flusso bancario.

Per garantire la persistenza di queste nuove entità senza compromettere lo schema originale del gestionale sono state introdotte specifiche tabelle applicative su Microsoft SQL Server, collegate ai dati standard dell'ERP mediante relazioni controllate e chiavi esterne.
Questa soluzione consente di mantenere separati il modello dati del gestionale e quello dell'estensione, preservando allo stesso tempo la congruità delle informazioni e il principio di non invasività che ha guidato l'intero progetto.

Ai fini dell'integrità del dato e della coerenza della logica applicativa, il design di queste entità segue precisi standard architetturali e aziendali, illustrati nel dettaglio nei prossimi paragrafi.

2.2.1 Identificazione e univocità

Ogni record è identificato internamente mediante una chiave primaria numerica generata automaticamente dal database.
Parallelamente, a livello applicativo è stato introdotto un vincolo di univocità logica basato sulla coppia costituita da numero e data della disposizione.

Tale vincolo consente di evitare duplicazioni accidentali e garantisce la corretta riconciliazione delle operazioni finanziarie durante l'intero ciclo di vita del flusso.

Al momento della generazione di una nuova disposizione di pagamento o distinta bancaria, il sistema valorizza automaticamente alcuni attributi fondamentali.
In particolare, la data del documento viene inizializzata con la data corrente di sistema, mentre il numero della disposizione viene determinato attraverso una numerazione progressiva giornaliera. Tale approccio consente di uniformare la gestione documentale e ridurre al minimo l'intervento manuale dell'operatore.

Contestualmente, il record viene creato nello stato iniziale *Provvisorio*, che rappresenta l'unico stato nel quale è consentita la modifica delle informazioni associate e delle operazioni aggregate.

2.2.2 Ciclo di vita e stati

Le entità introdotte sono caratterizzate da una macchina a stati che ne disciplina l'evoluzione operativa:

- **Provvisorio**: stato iniziale nel quale è possibile modificare metadati, associazioni e contenuto della disposizione;
- **Definitivo**: stato assunto successivamente alla generazione del file di invio conforme agli standard interbancari, che rende l'aggregato non più modificabile;

  Logica di rollback, da Definitivo a Provvisorio: Qualora un operatore debba correggere degli errori o variare le associazioni di una distinta in stato Definitivo (per esempio a causa di transazioni mancanti, importi non coperti o errori di compilazione), si è data la possibilità di "tornare indietro": Se l'utente tenta di associare/disassociare mandati o ricevute, il sistema mostra un messaggio informativo. Il messaggio avvisa l'utente che per procedere all'operazione è necessario che vada a riportare manualmente la distinta/disposizione in stato Provvisorio.
  Il cambio di stato svuota l'attributo fileName, invalidando di fatto il file generato in precedenza. L'utente può quindi modificare le transazioni interne alla distinta e procedere ad una nuova generazione del file, conforme alla configurazione aggiornata.

- **Inviato**: stato assunto dopo la trasmissione del file verso il gateway bancario, che conclude il ciclo operativo di andata della disposizione.

La seguente immagine 2.1 mostra attraverso uno state machine diagram il ciclo di vita dalla generazione all'invio.

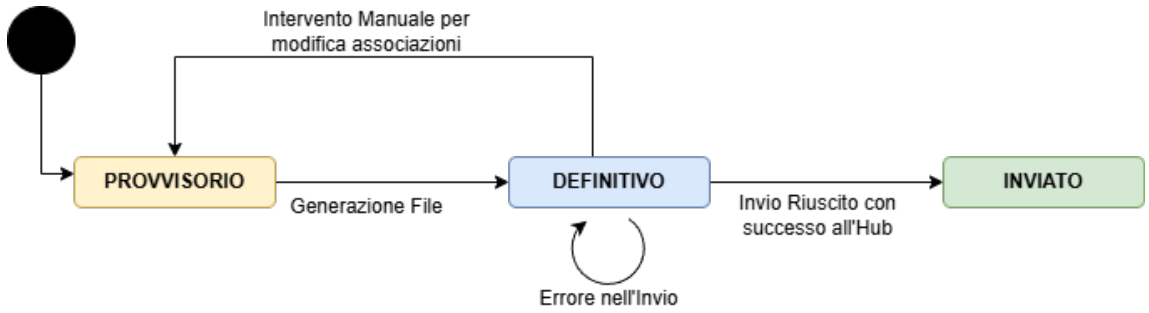

Figura 2.1: Diagramma di statdell'intercettazione dei flussi e gestione dei controlli preventivi [9].

2.2.3 Metadati operativi

Ogni entità memorizza informazioni necessarie alla gestione del ciclo finanziario, tra cui la banca assuntrice associata all'operazione e l'*Operation Result*, utilizzato per tracciare gli esiti restituiti dal sistema bancario a seguito delle operazioni di trasmissione, il cui tema verrà ampiamente approfondito nei capitoli finali.

In una successiva analisi esterna al periodo di tirocinio, dovuta a nuovi requisiti emersi dopo riunioni con il servizio esterno del gateway bancario, si è resa necessaria l'introduzione, negli oggetti complessi, dell'attributo *BankFlowType*, utilizzato per identificare la tipologia di flusso bancario da generare e trasmettere.

Le tipologie supportate sono gestite centralmente attraverso la tabella *BANKFLOWTYPE* e consentono di distinguere i diversi standard operativi previsti dal sistema. In particolare: per le *PaymentOrders* sono attualmente supportati i flussi SEPA Credit Transfer (bonifici SEPA) e Foreign Transfer (bonifici esteri); per le *BankReceipts* sono supportati i flussi SDD (SEPA Direct Debit) e RiBa (Ricevuta Bancaria).

L'introduzione di tale attributo risulta fondamentale poiché, pur condividendo il medesimo ciclo operativo, le diverse tipologie di flusso richiedono formati telematici e regole di generazione differenti.
Inoltre, la presenza dell'attributo *BankFlowType* rende il modello dati estensibile e predisposto all'introduzione di ulteriori standard bancari in future evoluzioni del sistema.
L'attributo risulta inoltre necessario per la distinzione a monte dell'associazione: una distinta creata per generare un file SDD (in formato XML) non potrà infatti mai contenere al suo interno anche transazioni RiBa (in formato TXT), data l'incompatibilità strutturale tra i due standard.

2.2.4 Arricchimento informativo

Per incrementare la tracciabilità delle operazioni, a ciascun record è possibile associare note descrittive, riferimenti ai documenti contabili di origine ed eventuali allegati, consentendo una gestione documentale completa a corredo della transazione finanziaria.

2.2.5 Layer di Presentazione e Interfaccia Utente

Il layer di presentazione è stato sviluppato in ambiente **Delphi** utilizzando, oltre alla *Visual Component Library* (VCL), anche e soprattutto i componenti proprietari messi a disposizione dal framework applicativo di Centro Software garantendo coerenza grafica e uniformità nei meccanismi di validazione dei dati.

La nuova sezione di interfaccia utente introdotta dal plugin si articola in due categorie principali di componenti:

- **Maschere di gestione** dedicate alla creazione, modifica e consultazione delle nuove entità applicative;
- **Griglie di visualizzazione** dedicate invece alla consultazione e al monitoraggio dei dati persistiti.

In particolare per la gestione delle disposizioni di pagamento e delle distinte bancarie sono state sviluppate le form `FPaymentOrders` e `FBankReceipts`, che consentono all'operatore di inserire e modificare i principali metadati associati al flusso finanziario.
Particolare attenzione è stata dedicata alla selezione della *Banca Assuntrice*.
A tale scopo è stato implementato un componente di ricerca collegato direttamente alle anagrafiche bancarie presenti nel database aziendale, consentendo all'utente di selezionare esclusivamente istituti di credito validi e censiti nel suo sistema.

Per la consultazione dei dati sono state invece sviluppate le griglie `GPaymentOrders` e `GBankReceipts`, attraverso le quali l'utente può visualizzare l'elenco delle disposizioni e delle distinte presenti nel sistema, applicando criteri di ordinamento e filtraggio coerenti con quelli già adottati dall'ERP, e soprattutto rimanendo sempre aggiornato sullo stato di avanzamento del flusso (sia di andata verso il sistema bancario esterno sua di ritorno).

Le schermate sviluppate, come brevemente anticipato, mantengono il medesimo paradigma di navigazione adottato da SAM ERP 2.0, riducendo l'impatto sul flusso operativo degli utenti e favorendo l'adozione delle nuove funzionalità in modo chiaro e guidato, senza richiedere attività formative specifiche fornite dalla consulenza applicativa e minimizzando la necessità di intervento del team di assistenza.

2.2.6 Logica di Integrazione e Intercettazione

Il lavoro svolto fino a questo momento si è basato su quel che si definisce *greenfield*, ovvero fatto da zero: nuove form, nuove tabelle, nuove griglie.
È ora che inizia la vera e propria integrazione con i processi *core* di SAM ERP 2.0, che ha rappresentato una delle sfide più rilevanti dell'intero progetto.
Il principale vincolo architetturale già ampiamente citato è dato dall'assenza di accesso al codice sorgente del gestionale, condizione che ha richiesto la progettazione di una soluzione in grado di estendere il comportamento dell'applicazione senza modificarne direttamente l'implementazione.

L'approccio adottato si basa sul principio di integrazione *non invasiva*: anziché sostituire o replicare le funzionalità esistenti del gestionale, il plugin si inserisce nei punti di estensione messi a disposizione dal framework applicativo, riutilizzando l'interfaccia e la logica già presenti all'interno del sistema.

Da un punto di vista architetturale, il modulo opera attraverso due meccanismi complementari, che permettono l'introduzione degli oggetti complessi descritti nelle sezioni precedenti:

- **L'iniezione dinamica dei componenti grafici (UI Injection)**, utilizzata per estendere dinamicamente le interfacce utente esistenti mediante l'aggiunta di nuovi componenti.
  La UI Injection consente al plugin di individuare a runtime specifiche maschere del gestionale e di arricchirle con componenti aggiuntivi sviluppati nel modulo di estensione.
  In questo modo nuove funzionalità vengono rese disponibili all'utente mantenendo inalterata l'esperienza operativa standard.

- **Redirect Event**, utilizzato per intercettare e integrare il comportamento degli eventi standard del gestionale, introducendo controlli, validazioni e funzionalità aggiuntive senza alterare il flusso nativo dell'applicazione.

2.3 L'importanza e l'uso sistematico del meccanismo di Redirect Event

L'integrazione del plugin all'interno di SAM ERP 2.0 si concretizza principalmente attraverso l'applicazione della tecnica di **Redirect Event**, la quale, a differenza di un tradizionale meccanismo di *override*, opera a runtime sostituendo dinamicamente il puntatore associato a un evento con un nuovo handler controllato dal plugin.
L'handler originale non viene eliminato, ma conservato e invocato programmaticamente, realizzando un modello di *Event Chaining*.
In questo modo, la logica introdotta dal plugin avvolge il comportamento nativo del gestionale senza alterarne né l'interfaccia né l'implementazione, declinando a livello applicativo i principi del pattern *Decorator*, secondo cui è possibile attribuire dinamicamente responsabilità aggiuntive ad un oggetto come alternativa flessibile alla sottoclassificazione [4].

Concretamente, la procedura interna sviluppata dal core aziendale per realizzare tale meccanismo espone una firma del tipo:

```
RedirectEvent(ComponenteX, EventoComponenteX, VecchioComportamento,
    NuovoComportamento);
```

dove *VecchioComportamento* rappresenta il riferimento all'handler nativo originariamente associato all'evento, mentre *NuovoComportamento* è la procedura definita nel plugin che verrà eseguita in sua sostituzione.
Il recupero del comportamento standard avviene in modo controllato all'interno della

nuova procedura, mediante un controllo di assegnazione che ne verifica l'esistenza prima di invocarlo:

```
if Assigned(VecchioComportamento) then
    VecchioComportamento(Sender, Field);
```

Questo controllo è ciò che rende possibile l'*Event Chaining*: il plugin può decidere se, quando e in quale punto della propria logica far proseguire il flusso verso il comportamento nativo, anziché sostituirlo integralmente.
La presenza della verifica *Assigned* garantisce inoltre la robustezza del meccanismo anche nei casi in cui l'evento originale non fosse stato precedentemente associato a un handler.

Sotto il profilo architetturale, questa intercettazione sistematica degli eventi permette di emulare la filosofia dei modelli *Publisher-Subscriber* che caratterizza i sistemi gestionali di ultima generazione: gli oggetti dell'applicazione base dichiarano dei publisher attivati in punti definiti del proprio flusso di esecuzione, mentre le estensioni esterne si limitano a dichiarare procedure subscriber, invocate dal dispatcher della piattaforma quando il publisher corrispondente viene attivato [5].

La dipendenza fluisce esclusivamente dal modulo di estensione verso l'evento pubblicato dal sistema base, il quale rimane strutturalmente agnostico rispetto all'esistenza stessa del plugin: è proprio questa direzione univoca della dipendenza, dal subscriber al publisher e non viceversa, a preservare l'isolamento tra applicazione base ed estensione [5].

La sicurezza nei cicli di aggiornamento (*upgrade safety*) deriva da un meccanismo più specifico: quando la piattaforma di base viene aggiornata, le estensioni che si limitano a sottoscrivere eventi non vengono toccate, poiché è un livello di compatibilità dedicato a validare che le interfacce evento da cui l'estensione dipende non siano state alterate nella nuova versione [5]. Applicato al caso di SAM ERP 2.0, questo si traduce in un flusso operativo in cui il plugin intercetta il trigger, inietta le logiche necessarie — quali la validazione formale dell'IBAN o il raggruppamento automatico dei mandati — e ripristina il flusso standard, preservando l'integrità transazionale del core ERP.

Dal punto di vista operativo, il plugin interviene secondo due modalità distinte:

- **Pre-processing**: esecuzione di controlli preliminari, validazioni di coerenza e operazioni di contestualizzazione prima che venga attivata la logica standard del gestionale;
- **Post-processing**: esecuzione di attività aggiuntive successive all'elaborazione nativa, quali l'aggiornamento degli stati applicativi, la sincronizzazione dei dati e le operazioni di persistenza supplementari.

Nel contesto specifico del progetto, il meccanismo di Redirect Event è stato applicato con successo in più momenti distinti approfonditi nelle sezioni successive.

La Figura 2.2 riporta un diagramma di attività ad alto livello di astrazione sul funzionamento del meccanismo e su dove esso si inserisce nel flusso applicativo.

Il diagramma non considera le eccezioni che potrebbero essere sollevate nei singoli casi d'uso, limitandosi a fornire una visione d'insieme generale del meccanismo.

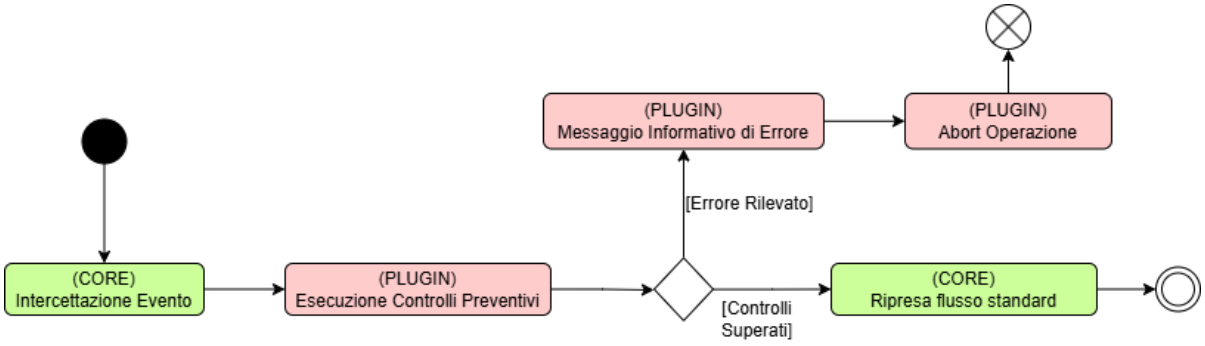

Figura 2.2: Diagramma di attività dell'intercettazione dei flussi e gestione dei controlli preventivi [10].

2.3.1 Estensione del filtraggio e contestualizzazione dei dati

All'apertura delle maschere standard del gestionale relative ai singoli mandati di pagamento e alle ricevute bancarie, osserviamo una form di filtraggio che consente all'operatore di circoscrivere i record visibili secondo criteri predefiniti (banca, data, stato contabile).
L'integrazione ha arricchito questa form mediante l'inserimento dinamico di un componente aggiuntivo che introduce un ulteriore livello di filtraggio basato sulla **Disposizione/Distinta** di riferimento.

La lista delle disposizioni disponibili per il filtraggio non è statica, ma viene popolata dinamicamente in funzione del contesto operativo corrente.
In particolare, per le ricevute bancarie, la selezione è vincolata alla tipologia di flusso attiva: qualora l'operatore selezioni la modalità *RiBa* tramite il controllo radio button presente nella maschera, l'elenco presenterà esclusivamente le distinte di tipo RiBa; analogamente, la selezione della modalità *SDD* aggiorna dinamicamente la lista per mostrare unicamente le distinte SEPA Direct Debit.
Questo comportamento garantisce la coerenza semantica del filtraggio, impedendo all'operatore di associare titoli a distinte di tipologia incompatibile.
Inoltre, le distinte già trasmesse al gateway bancario — il cui stato è *Inviato* — sono escluse dall'elenco, eliminando alla radice il rischio di doppia elaborazione di flussi già processati.

Per garantire il corretto funzionamento di questo meccanismo, il plugin intercetta due eventi chiave:

- `BtnOkOnClick`: l'evento di conferma del filtro viene intercettato per persistere in memoria la disposizione selezionata dall'utente. Tale informazione definisce il *perimetro operativo* della sessione corrente, vincolando tutte le operazioni successive al contesto della disposizione scelta.

- `BeforeOpenGrid`: l'intercettazione di questo trigger, eseguita immediatamente prima del popolamento della griglia dati, consente di iniettare una clausola `WHERE`

aggiuntiva nella query SQL sottostante. La clausola forza il sistema a visualizzare esclusivamente i titoli coerenti con la disposizione selezionata e non ancora trasmessi al gateway bancario.

Questo doppio livello di intercettazione garantisce che la griglia operativa rifletta sempre uno stato consistente rispetto al ciclo di vita della disposizione, prevenendo anomalie contabili derivanti da selezioni incongrue.

2.3.2 Manipolazione della Toolbar e logica di Associazione Massiva

Per abilitare l'aggregazione logica dei titoli nelle entità complesse introdotte dal plugin, la toolbar standard del gestionale è stata estesa con un nuovo Button, in modo da richiamare la nuova maschera `ASSOCIA`, sviluppata all'interno del modulo di estensione.

Questa form implementa una logica di **selezione multipla** basata su *checkbox* sulla griglia dati, consentendo all'operatore di operare in modalità massiva su insiemi eterogenei di pendenze contabili.
L'interfaccia espone due operazioni distinte:

- **Associazione massiva**: permette di collegare univocamente un insieme di record selezionati a una specifica *PaymentOrder* o *BankReceipt*.
  Il vincolo di unicità è garantito a livello applicativo: un titolo non può essere associato contemporaneamente a più disposizioni, preservando l'integrità del flusso finanziario, di conseguenza, se uno tra i titoli selezionati era già associato adun altra disposizione/distinta, l'associazione verrrà sovrascritta con quella corrente.

- **Disassociazione massiva**: consente di sciogliere il legame tra un insieme di titoli e la loro disposizione corrente, ripristinando lo stato di *pendenza* del record originale nell'ERP e rendendolo nuovamente disponibile per una successiva aggregazione.

2.3.3 Feedback visivo e monitoraggio dello stato (`DataChange`)

Al fine di migliorare l'efficienza operativa e migliorare l'esperienza utente, il pannello informativo posizionato sopra la griglia dati è stato esteso con le informazioni contestuali aggiuntive relative ai nuovi oggetti aggreganti.

Il meccanismo si basa sull'intercettazione dell'evento `OnDataChange` della sorgente dati (`DataSource`) associata alla griglia: ogni volta che il cursore si sposta su un record differente, il plugin ricalcola il contenuto del `Panel` descrittivo e ne aggiorna la visualizzazione in modo sincrono con l'interfaccia del gestionale.

Oltre ai metadati standard esposti da SAM ERP 2.0, ora sono presenti:

- Il riferimento alla **Disposizione/Distinta** associata al titolo corrente, permettendo all'operatore di verificare immediatamente il contesto di aggregazione senza necessità di aprire maschere di dettaglio;

- Un **indicatore di origine** (limitatamente ai mandati di pagamento), che specifica se il record deriva dal ciclo contabile strutturato dell'ERP oppure è stato introdotto manualmente tramite l'entità `ManualPaymentOrders`. Questa distinzione è critica per la riconciliazione contabile, poiché i mandati manuali non sono collegati a un documento contabile d'origine nel sistema e richiedono una validazione aggiuntiva prima della trasmissione.

2.3.4 Override della generazione del file e validazione dei vincoli

Infine,il punto di maggiore criticità è rappresentato dalla sovrascrittura del comando nativo di generazione del file telematico che verrà poi inviato al gestore bancario.

Prima di procedere alla creazione del file, il plugin *sospende* il flusso standard per eseguire una serie di **validazioni bloccanti**:

- **Integrità dell'aggregato**: il sistema verifica che tutti i record selezionati per la generazione siano associati alla medesima Disposizione/Distinta.
  Qualora venga rilevata un'incongruenza — ad esempio la presenza di titoli afferenti a disposizioni differenti, oppure record non associati a nessuna disposizione — il processo viene interrotto e all'utente viene restituito un messaggio di errore descrittivo.
  Questo controllo previene la generazione di file ibridi, che risulterebbero non conformi agli standard interbancari e verrebbero rifiutati dal gateway banncario nel momento dell'invio.

- **Completezza dell'aggregato**: il sistema garantisce la perfetta corrispondenza tra i record selezionati nell'interfaccia utente e l'entità logica della Disposizione presente a database.
  In particolare, il plugin verifica che, se per una determinata Disposizione Y risultano associati a sistema N titoli, l'utente ne abbia selezionati esattamente N per l'invio.
  Qualora venisse rilevata una selezione parziale (ovvero X record selezionati dove $X < N$), il processo viene bloccato.
  Questa validazione è fondamentale per assicurare che il flusso inviato al gateway sia esaustivo e che non rimangano pendenze "orfane" all'interno di SAM.
  Si evita così il rischio di disallineamenti tra lo stato dei pagamenti sul gestionale e le operazioni effettivamente trasmesse al circuito bancario.

- **Forzatura della Banca Assuntrice**: la logica di generazione viene modificata per ignorare le banche eventualmente indicate sui singoli titoli e imporre, in modo sistematico, la banca definita nella testata della Disposizione/Distinta.
  Questo meccanismo garantisce la quadratura del flusso finanziario e l'aderenza al requisito RF8, eliminando l'inconsistenza che si produrrebbe nel caso in cui titoli appartenenti alla stessa disposizione risultassero instradati su istituti di credito differenti.

Solo al superamento di tutte le validazioni il plugin consente la prosecuzione del flusso originale, delegando nuovamente al core del gestionale la generazione materiale del file

secondo i propri algoritmi nativi.

Contestualmente all'operazione di generazione file viene aggiornato lo stato della disposizione da *Provvisorio* a *Definitivo* (requisito RF9), congelando l'aggregato e impedendone ulteriori modifiche prima della trasmissione.

Questa serie di intercettazioni trasforma l'ERP da un sistema passivo di archiviazione a un **motore attivo di gestione finanziaria**, in cui ogni azione è validata, contestualizzata e resa coerente rispetto agli oggetti complessi introdotti dal plugin e ai vincoli imposti dal gateway bancario.

2.4 Valutazione e Limiti dell'approccio scelto

Analizzando in modo critico il lavoro svolto, possiamo individuare e conseguentemente analizzare i limiti strutturali dell'approccio utilizzato, così da fornire un quadro completo delle implicazioni progettuali e delle direzioni di miglioramento identificabili.

Fragilità rispetto agli aggiornamenti del framework
Il meccanismo di Redirect Event si basa sulla sostituzione a runtime dei puntatori a funzione degli eventi esposti dal framework di Centro Software.
Sebbene questo approccio garantisca l'isolamento dal codice sorgente, introduce una dipendenza implicita dalla *firma* degli eventi intercettati: qualora un aggiornamento del gestionale dovesse modificare il nome, i parametri o la semantica di un evento, il plugin cesserebbe di funzionare correttamente, richiedendo un intervento di manutenzione mirato.
Nonostante ciò, questo rischio è stato mitigato attraverso una strategia rigorosa: l'utilizzo esclusivo di eventi e interfacce standard espressamente documentati nell'API pubblica del framework, evitando qualsiasi dipendenza da logiche interne non ufficiali dell'ERP.

Il modulo di adattamento *ExFun* e la dipendenza dal file *exfun.ini*
Per centralizzare l'intera logica di estensione e disaccoppiarla dal resto del sistema, è stato sviluppato un modulo dedicato denominato `ExFun`. Questo modulo funge da unico punto di adattamento e di iniezione tra il plugin bancario e il core dell'ERP.

L'identificazione delle specifiche maschere su cui questo modulo deve intervenire è delegata a un file di configurazione esterno, denominato *exfun.ini*, che funge da vero e proprio registro di attivazione. Se da un lato questa scelta massimizza la flessibilità — consentendo di mappare nuove form e abilitare l'estendibilità della UI senza richiedere la ricompilazione del codice — dall'altro introduce un vincolo operativo.
L'assenza, la corruzione o un errore di sintassi in questo file impedisce al modulo `ExFun` di sapere su quali maschere attivarsi. In questo scenario, il gestionale continuerebbe a caricare esclusivamente la sua interfaccia standard, rendendo di fatto invisibili i nuovi componenti grafici progettati per l'integrazione bancaria, senza però sollevare eccezioni o errori bloccanti per l'ERP.

Una soluzione parziale per aumentare la robustezza del sistema consiste nell'introduzione di un meccanismo di validazione all'avvio del modulo, che verifichi l'integrità del file sul filesystem e notifichi l'amministratore in caso di anomalie di lettura.

Difficoltà di testing automatizzato

La natura *black-box* del gestionale, se da un lato ha guidato le scelte architetturali descritte nei capitoli precedenti, dall'altro pone un vincolo importante relativo alla verificabilità del plugin: non essendo possibile istanziare il core di SAM ERP 2.0 in un ambiente isolato o headless, non è praticabile predisporre una suite di test automatizzati che validino il comportamento del Redirect Event a fronte di una nuova versione del gestionale.
La verifica del corretto funzionamento rimane pertanto un'attività in larga parte manuale, condotta in primo luogo da me durante l'implementazione e in una fase successiva dal team di collaudo di Centro Software.

Nonostante ciò, questa modalità di verifica comporta non solo un allungamento dei tempi di validazione, ma soprattutto un rischio di regressione difficile da eliminare: chi collauda il plugin conosce già a fondo il funzionamento previsto del sistema e tende implicitamente a testarlo seguendo i percorsi d'uso attesi, lasciando scoperti gli scenari anomali o le sequenze di interazione meno convenzionali.
Un cliente che, nell'uso quotidiano, si muove al di fuori di questi percorsi può quindi imbattersi in comportamenti anomali che il collaudo manuale, per sua stessa natura, non è strutturato per individuare. Una suite di test automatizzati, pur con i limiti di praticabilità già descritti, ridurrebbe questo margine di rischio introducendo una copertura sistematica indipendente dall'esperienza pregressa del collaudatore.

Capitolo 3

Integrazione API-centric con il Gateway Bancario

Il presente capitolo si discosta dal lavoro di integrazione svolto durante il periodo di tirocinio, e riguarda il livello di integrazione più esterno dell'architettura sviluppata: la comunicazione diretta tra il plugin e il gateway bancario, realizzata attraverso un canale API-centric che sostituisce integralmente il precedente modello di trasmissione manuale dei file.

Se nei capitoli precedenti è stata illustrata la realizzazione del plugin nel suo complesso, e di come si inserisce nei flussi interni di SAM, ora siamo ad un punto di equilibrio: tutte le componenti necessarie alla gestione delle disposizioni e delle distinte sono state realizzate e integrate nel sistema esistente.

Proprio a partire da questo punto di equilibrio, ora il focus dei prossimi capitoli si sposta verso il confine esterno del sistema.
Nelle griglie degli oggetti complessi — dove sono elencate tutte le disposizioni e le distinte gestite dal plugin — si apre infatti l'interfacciamento con il servizio bancario esterno.

La Figura 3.1 sotto riportata, mostra una visione d'insieme della comunicazione appena introdotta, articolata nei tre attori coinvolti — il plugin all'interno di SAM ERP 2.0, il gateway e il sistema interbancario — e nei due flussi che la caratterizzano: l'invio delle disposizioni/distinte e il ritorno dei relativi esiti. I paragrafi successivi approfondiscono nel dettaglio ciascuna delle fasi qui rappresentate.

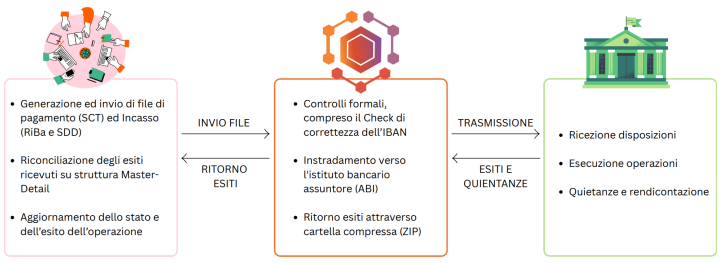

Figura 3.1: Schema riassuntivo ad alto livello di astrazione del ruolo del gateway bancario [11].

3.1 Introduzione al servizio messo a disposizione dall'hub bancario

Le specifiche funzionali e architetturali descritte nel presente paragrafo e nei successivi sono state ricostruite a partire dalla documentazione tecnica e commerciale fornita dall'intermediario bancario **WikiSoftware** [2], integrata dalle informazioni emerse nel corso delle riunioni di analisi svolte con il referente del servizio.
Tale documentazione, di natura privata e sensibile in quanto contenente specifiche tecniche e commerciali proprietarie, non è consultabile pubblicamente e non viene pertanto inclusa tra le fonti bibliografiche di questo lavoro.

Il servizio fornito dalla piattaforma bancaria ha lo scopo di mettere in comunicazione, in maniera semplificata e accentrata, applicativi gestionali e di tesoreria attraverso un insieme di API esposte dall'hub.
In base alla tipologia di flusso bancario trattato, l'infrastruttura offre due macro-tipologie di web service:

- **Send**: invocati per inviare al gateway i flussi dispositivi da veicolare sul circuito interbancario attraverso i canali comunicativi dei clienti;
- **Message**: invocati per ricevere dal gateway i flussi informativi scaricati dal circuito interbancario.

Attualmente, il sistema gestionale SAM supporta, tramite l'integrazione in oggetto, i seguenti flussi finanziari e i relativi formati di interscambio:

- **Bonifici SEPA (SEPA Credit Transfer):** formati CBI XML e ISO 20022;
- **Addebiti diretti SEPA (SEPA Direct Debit):** formati CBI XML e ISO 20022;
- **Ricevute Bancarie (Ri.Ba.):** standard nazionale in formato CBI TXT.

Dal punto di vista architetturale, il gateway si interpone tra il gestionale aziendale e il sistema bancario, esponendo i propri servizi tramite interfacce API REST — i cosiddetti endpoint — attraverso cui il plugin comunica direttamente con la piattaforma bancaria. L'infrastruttura opera inoltre in conformità agli standard ISO 27001:2022, che regolamenta la gestione della sicurezza delle informazioni, e ISO 9001:2015, relativo alla gestione della qualità dei processi — requisiti particolarmente rilevanti in un contesto transazionale finanziario.

3.2 Architettura del modulo di integrazione

La logica di comunicazione con il gateway bancario è stata incapsulata in una libreria dinamica (`DLL`) indipendente, caricata a runtime dall'ambiente del plugin.
Questa scelta progettuale non risponde esclusivamente a una necessità tecnica, ma riflette una precisa strategia di gestione del rischio architetturale: isolare in un componente autonomo e sostituibile tutta la logica di integrazione con il servizio esterno significa che eventuali evoluzioni delle API della piattaforma bancaria — o la necessità di supportare

gateway bancari alternativi in futuro — possono essere recepite aggiornando esclusivamente la libreria, senza toccare nient'altro.

Il secondo vantaggio è di natura operativa: il modello a DLL garantisce la compatibilità con le installazioni basate sulle versioni precedenti del gestionale, che possono caricare la libreria senza richiedere alcuna modifica.
In un parco clienti eterogeneo come quello di Centro Software, dove coesistono installazioni sulla serie 5.x e sulla serie 6.x, questa proprietà si traduce direttamente in un contenimento dei costi di deployment e manutenzione: un unico artefatto distribuibile copre l'intera base installata, eliminando la necessità di gestire rami di rilascio paralleli per la componente di integrazione bancaria.

L'interfaccia esposta dalla DLL è volutamente minimale, contiene infatti solo le funzioni strettamente necessarie alla comunicazione.
Per la fase di trasmissione è stata definita un'unica funzione che in un solo momento gestisce l'autenticazione tramite token e l'invio del file al gateway.
Questa scelta di design riduce il coupling tra il modulo di integrazione e il resto del sistema: il plugin non conosce i dettagli del protocollo di autenticazione né la struttura interna della chiamata HTTP, ma si limita a invocare la funzione con i parametri applicativi necessari, delegando interamente alla DLL la responsabilità dell'avvenuta comunicazione.

3.3 Provisioning, autenticazione e modello di licensing

Prima di poter trasmettere flussi bancari tramite il gateway, ciascuna installazione deve essere censita nel sistema bancario esterno mediante un processo di provisioning che associa l'azienda ad un token identificativo univoco.
Tale token non è mai visibile all'operatore nell'interfaccia utente: viene gestito internamente dal sistema e utilizzato in modo trasparente a ogni chiamata API.

Questo meccanismo garantisce che le credenziali di accesso al sistema bancario non siano mai nè visualizzabili nè modificabili dall'operatore aziendale, e tanto meno esposte in chiaro nei log applicativi, riducendo la superficie di attacco e conformando il sistema alle *best practice* di sicurezza in ambito Fintech.

Importante è sottolineare che dal punto di vista della governance aziendale, la release 6.13 di SAM ERP 2.0 introduce una netta separazione tra le funzionalità core di gestione documentale e i servizi di connettività finanziaria a valore aggiunto.
Questa ripartizione si traduce in un modello di licensing modulare:

- **Funzionalità Core (Incluse):** Il motore di modellazione degli oggetti complessi (*PaymentOrders* e *BankReceipts*), i meccanismi di associazione massiva e la generazione del file di invio conforme agli standard interbancari (ovvero tutto il lavoro svolto nella prima parte di tirocinio e descritto precedentemente) sono inclusi nel canone standard di Centro Software.

- **Integrazione API Hub Bancario (Pacchetto Aggiuntivo):** L'attivazione del canale diretto via API REST verso l'hub finanziario richiede un accordo commerciale specifico.
  Questo modulo abilita il provisioning del token aziendale e include un pacchetto di chiamate API preconcordato.

Sotto il profilo dei costi operativi per le PMI, la convenienza del servizio è legata alle partnership strategiche in essere.
Nello specifico, l'infrastruttura bancaria opera in sinergia nativa con una specifica banca; le aziende che selezionano tale istituto come *Banca Assuntrice* beneficiano di tariffe agevolate, pur mantenendo la totale copertura verso qualsiasi altro istituto bancario nazionale ed europeo a tariffe standard meno competitive.

Implementazione del controllo di abilitazione (Feature Flagging)

Per supportare questo modello di business a livello di codice senza incrementare il debito tecnico o creare rami di compilazione separati, la logica di *UI Injection* descritta nella Sezione 2.2.3 interroga il modulo delle licenze aziendali dell'ERP prima di completare il rendering della toolbar operativa.

Il controllo viene mediato da un flag booleano (bit di validità di licenza) persistito nel database aziendale. Il flusso di inizializzazione si articola secondo la seguente logica:

1. All'attivazione della maschera della griglia (`GPaymentOrders` o `GBankReceipts`), il plugin richiama il metodo di controllo abilitazione integrato nel modulo `ExFun`.

2. Se il bit di validità è impostato su `True` (contratto attivo con l'intermediario), il sistema completa l'iniezione della UI inserendo i pulsanti di invio (`BtnSend`) e di ritorno (`BtnReturn`) nella toolbar, ed impostandoli in stato abilitato (`Enabled = True`), pronti per scatenare le relative chiamate API al *click*.

3. Se il bit di validità è impostato su `False`, il sistema adotta una strategia silente o di *visual restriction*: i pulsanti di invio e ritorno vengono impostati visivamente come non cliccabili (`Enabled = False`).
   In questo modo, l'interfaccia inibisce l'azione all'origine, forzando l'utente a usufruire unicamente del flusso di salvataggio manuale del file su disco incluso nel pacchetto base.

3.3.1 Flusso operativo di trasmissione dei file di invio

L'invio del flusso bancario è innescato dall'operatore contabile tramite il pulsante dedicato aggiunto alla toolbar delle griglie `GPaymentOrders` e `GBankReceipts`.
Il comando è reso disponibile esclusivamente sulle disposizioni il cui stato è *Definitivo*, ovvero per le quali è già stato generato il file di invio telematico e il cui percorso locale è mappato nell'attributo `FileName`.

Prima di avviare la trasmissione del payload, il plugin gestisce in modo trasparente l'autenticazione con l'hub finanziario tramite protocollo HTTPS. La sicurezza dell'accesso è garantita dallo standard **OAuth 2.0** mediante il *grant type* di tipo *client_credentials*,

ideale per comunicazioni *machine-to-machine*.
Utilizzando le **credenziali riservate** fornite in fase di provisioning (`client_id` e `client_secret`), il sistema interroga l'endpoint di autenticazione per ottenere un *Bearer Token* (JSON Web Token) a scadenza temporale.
Questo token viene memorizzato temporaneamente in memoria dal plugin e iniettato nell'header HTTP `Authorization` di ogni successiva chiamata dispositiva, venendo riutilizzato per l'intera durata della sua validità ed evitando così chiamate di autenticazione ridondanti, senza richiedere alcun inserimento di credenziali da parte dell'utente.

Una volta stabilito il canale sicuro, viene effettivamente eseguito l'invio, sempre grazie alla funzione esposta dalla DLL, la quale richiede alcuni parametri per costruire le chiamata:

- **Percorso del file XML:** il riferimento locale del flusso telematico generato nella fase di override descritta nella Sezione 2.3.4;
- **ID Azienda:** l'identificativo univoco dell'azienda, utilizzato per instradare correttamente il flusso verso il conto aziendale di riferimento;
- **ABI:** il codice identificativo dell'istituto bancario assuntore, derivato dalla testata della disposizione;
- **Tipo di flusso:** la tipologia del flusso bancario da trasmettere (*SEPA Credit Transfer*, *SDD* o *RiBa*), che determina il formato atteso dal gateway e le relative regole di validazione.

Il flusso di comunicazione logica, le interazioni tra i componenti del sistema e la gestione delle risposte dell'endpoint esterno sono schematizzati nel diagramma di sequenza di Figura 3.2.
Il modello eredita la struttura logica descritta nella documentazione ufficiale dell'intermediario, raffinandola e specializzandola in base alla specifica implementazione del plugin per riflettere la reale interazione tra l'interfaccia utente del gestionale e la DLL di comunicazione.

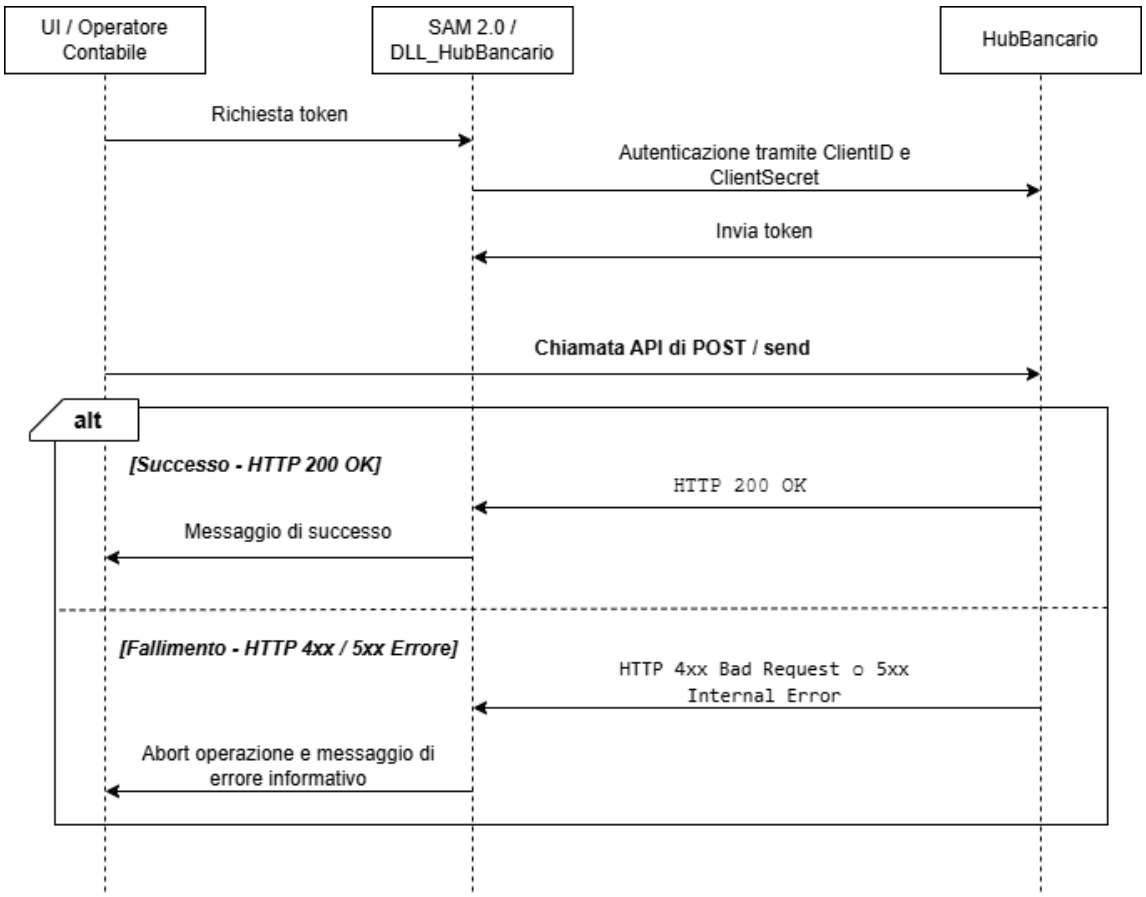

Figura 3.2: Diagramma di sequenza dell'autenticazione tramite Client Credentials e trasmissione del payload, raffinato in base alle specifiche di implementazione e documentazione ufficiale [12]).

3.3.2 Firma dispositiva tramite PIN dinamico per le disposizioni di pagamento

Per le operazioni finanziarie che comportano una movimentazione in uscita di fondi — nello specifico i bonifici *SEPA Credit Transfer* (SCT) — il gateway bancario, prima dell'effettivo invio del documento,impone un ulteriore livello di sicurezza: la **firma dispositiva tramite PIN dinamico**.

A differenza del token di accesso automatico gestito dal sistema, questa procedura richiede un'azione consapevole da parte dell'operatore e si articola in due fasi distinte coordinate dal plugin:

1. **Richiesta e generazione dell'OTP:** Il plugin effettua una chiamata `POST` all'endpoint `/sendPin` trasmettendo l'identificativo aziendale e l'indirizzo email dell'operatore autorizzato censito a sistema. Il gateway genera un codice temporaneo (valido fino alla mezzanotte del giorno corrente), lo invia via email all'utente e restituisce al plugin un `transactionId` univoco per tracciare la sessione dispositiva.

2. **Validazione e firma crittografica del payload:** L'operatore inserisce il PIN ricevuto nell'interfaccia grafica del plugin. Al momento del clic sull'invio definitivo,

il sistema calcola l'hash crittografico (SHA256) del codice e lo utilizza per firmare il file binario della distinta tramite algoritmo **HMAC256**.
Il payload così cifrato e il `transactionId` vengono inviati all'endpoint di trasmissione definitivo per completare la disposizione.

Questo approccio a doppio fattore garantisce che, per i flussi dispositivi ed esposti a rischio come i bonifici, nessuna transazione possa essere completata senza l'accesso fisico alla casella di posta elettronica autorizzata, separando la comunicazione automatica dell'ERP dall'approvazione formale e manuale dell'operazione contabile.

3.4 Standard e struttura dei file dei flussi generati

L'architettura del plugin è progettata per garantire una netta separazione tra la **logica di aggregazione** delle operazioni e la **logica di serializzazione** dei flussi.
Le entità applicative introdotte (*PaymentOrder* e *BankReceipt*) costituiscono un modello dati astratto e indipendente dal formato finale di trasmissione.

Solo nella fase finale del processo, tali informazioni vengono trasformate nel tracciato tecnico richiesto dal circuito bancario.
In particolare troviamo da un lato le Ri.Ba. (Ricevuta Bancarie) che si basano ancora sul tradizionale e storico standard nazionale con flusso **IB** con tracciato TXT a lunghezza fissa, mentre dall'altro lato i SEPA Credit Transfer con flusso **(SCT)** e i SEPA Direct Debit con flusso **(SDD)** entrambi basati sullo standard internazionale **XML ISO 20022**.

Questa distinzione riflette l'evoluzione dei sistemi di pagamento: mentre i flussi SEPA adottano formati strutturati e gerarchici, le Ri.Ba. utilizzano ancora un modello posizionale derivato dagli standard interbancari storici.

3.4.1 Flussi Ri.Ba.

Le ricevute bancarie seguono lo standard definito dal Consorzio CBI (Corporate Banking Interbancario). Il file prodotto è un documento testuale (*flat file*) a lunghezza fissa, in cui ogni informazione è identificata esclusivamente dalla sua posizione all'interno della riga (offset).

La struttura del flusso è organizzata in modo sequenziale:

1. **Record di testa (IB):** contiene i dati identificativi del mittente e le informazioni generali della distinta.
2. **Corpo del flusso:** composto da uno o più blocchi di record relativi alle singole disposizioni d'incasso.
3. **Record di coda (EF):** chiude il file riportando i dati di quadratura, come il numero totale delle disposizioni e l'importo complessivo trasmesso, garantendo l'integrità formale della spedizione.

A differenza dei moderni formati XML, il significato dei dati non è esplicitato da tag descrittivi; pertanto, anche una minima deviazione nel posizionamento dei caratteri può compromettere l'elaborazione del flusso da parte dei sistemi bancari.

3.4.2 Flussi SEPA (SCT e SDD)

Per i bonifici e gli addebiti diretti, il plugin genera documenti XML conformi allo standard **ISO 20022**. Tale modello garantisce una maggiore estensibilità e una validazione sintattica rigorosa tramite schemi **XSD**. Nello specifico:

- Per i bonifici (SCT) viene utilizzato il messaggio **pain.001** (*Payment Initiation* — il formato standard per l'inizializzazione di un ordine di pagamento);
- Per gli addebiti diretti (SDD) viene utilizzato il messaggio **pain.008** (*Direct Debit Initiation* — il formato standard per l'inizializzazione di un addebito diretto).

Entrambi i formati includono tag descrittivi per identificativi univoci, coordinate IBAN, dati delle controparti e causali, facilitando la successiva riconciliazione degli esiti descritta nel capitolo successivo, la cui efficacia dipende direttamente dalla qualità dei riferimenti trasmessi in questa fase.

3.5 Gestione della risposta e consistenza applicativa

Al termine della trasmissione, il plugin elabora la risposta restituita dal gateway bancario per allineare lo stato delle entità locali all'esito dell'operazione.

3.5.1 Esito Positivo

In presenza di un codice HTTP di successo (**200**), il file è considerato acquisito correttamente dal sistema bancario. Il plugin esegue quindi le seguenti azioni:

- **Avanzamento di stato:** La disposizione passa da *Definitivo* a *Inviato*.
- **Immutabilità:** Una volta nello stato *Inviato*, l'aggregato non è più modificabile, non è pertanto più possibile associare o rimuovere documenti, né rigenerare il file.
- **Tracciabilità:** Vengono registrati l'utente operatore e il timestamp della trasmissione.
- **Predisposizione Riconciliazione:** Viene creata una voce nelle tabelle di raccordo, che fungerà da base per l'elaborazione dei flussi di ritorno.

3.5.2 Esito Negativo

Qualora il gateway restituisca un errore (classi HTTP **4xx** o **5xx**), il plugin mantiene la disposizione nello stato *Definitivo*.
L'errore viene interpretato e presentato all'utente tramite una segnalazione parlante, permettendo di correggere eventuali anomalie anagrafiche o di configurazione direttamente nell'ERP.
Questo approccio evita di dover ricostruire l'intera distinta, consentendo di ripetere l'invio non appena risolta la causa del blocco.

Capitolo 4

Flussi di ritorno e gestione degli esiti: la chiusura del ciclo informativo

Una volta inviati i flussi verso il sistema bancario, il processo di integrazione non può pensarsi completo finché non viene affrontata la questione speculare: la ***ricezione degli esiti***.

In un'ottica di digitalizzazione dei processi aziendali, il valore reale di un'integrazione non risiede soltanto nella capacità di trasmettere dati, ma nella possibilità di chiudere il ciclo informativo, ricevendo in modo strutturato e automatico le risposte che la controparte — in questo caso il sistema bancario — genera una volta conclusa l'elaborazione.

Il servizio *Message* esposto dalle API della controparte assolve precisamente a questa funzione: il recupero dei file restituiti dalla banca.
Dal punto di vista concettuale, si tratta di un meccanismo di ***asynchronous callback***, nel quale il mittente originale del flusso assume temporaneamente il ruolo di destinatario, interrogando periodicamente il sistema per verificare l'avanzamento delle proprie richieste.

Questa scelta architetturale — preferire il ***polling asincrono*** rispetto a una notifica in tempo reale — non è un dettaglio implementativo, ma riflette una tendenza strutturale nell'evoluzione dei sistemi di integrazione aziendale. Va precisato che tale approccio è anche un vincolo imposto dall'hub bancario esterno, il quale, non predisponendo di meccanismi di notifica push, rende il polling l'unico modello di ricezione disponibile.
Ciò nonostante, questa modalità si dimostra pienamente adeguata al contesto operativo, poiché la natura stessa dei flussi bancari — caratterizzati da latenze di elaborazione strutturalmente elevate — non richiede una sincronia in tempo reale tra le parti.

Sul piano teorico, Hohpe e Woolf avevano già posto la messaggistica asincrona al centro dei pattern fondamentali per l'integrazione tra applicazioni distribuite [7]; Ritter, May e Rinderle-Ma hanno poi osservato come questa preferenza si sia ulteriormente consolidata con l'affermarsi degli ecosistemi ibridi, nei quali applicazioni on-premise e applicazioni cloud devono cooperare senza che sia garantita la sincronia tra le parti [8].

Il sistema bancario, per sua natura distribuito e con latenze di elaborazione strutturalmente elevate, rappresenta un caso emblematico di questo scenario.

Un ulteriore elemento di complessità riguarda la varietà dei formati con cui il sistema deve operare nel suo complesso: sebbene ciascun esito venga restituito nello stesso formato del flusso di invio a cui si riferisce, il plugin deve comunque essere in grado di interpretare e normalizzare diversi tipi di codifiche che convivono nello stesso ecosistema rispecchiando stratificazioni storiche nei sistemi di comunicazione interbancaria.
Anche questa eterogeneità non è un'anomalia locale, ma una manifestazione di una sfida sistemica ampiamente documentata: Ritter, May e Rinderle-Ma identificano proprio nella **varietà dei formati** — insieme alla crescita dei volumi e alla velocità di scambio — una delle tre dimensioni critiche che i sistemi di integrazione moderni devono saper gestire [8].
Progettare un sistema capace di ricevere e normalizzare formati diversi senza esporre questa complessità ai livelli applicativi superiori è una scelta architetturale strategica: permette infatti di isolare il core del gestionale dalle specifiche dei singoli istituti bancari, garantendo al contempo un'esperienza utente fluida, uniforme e centralizzata.

4.1 La struttura dati per la riconciliazione

Il problema più delicato nella gestione dei flussi di ritorno non è tanto tecnico quanto concettuale: non è la lettura del file il problema, ma **l'organizzazione dei dati raccolti**.
Proprio per questo, l'intero focus della discussione è stato orientato attorno a una domanda precisa: *come organizzare i dati in modo da poter ricondurre rapidamente ogni esito ricevuto alla corrispondente disposizione inviata in precedenza?*
Questa esigenza di riconciliazione è centrale in qualsiasi processo di integrazione tra sistemi eterogenei e costituisce uno dei nodi critici della digitalizzazione dei flussi finanziari aziendali.

La trattazione che segue si concentra in particolare sui flussi attivi di incasso — RiBa e SDD — che hanno costituito l'ambito diretto del mio intervento progettuale, ma che rappresentano anche la porzione più complessa del problema dal punto di vista della gestione degli esiti: a differenza dei flussi di pagamento, infatti, i flussi di incasso possono generare insoluti, situazioni che richiedono una gestione granulare e attenta a livello di singola transazione.
La medesima architettura concettuale è stata successivamente estesa anche ai flussi di pagamento SCT, le cui implicazioni gestionali risultano più semplici rispetto a quelle dei flussi attivi, e di cui non si entra qui nel dettaglio implementativo.

La soluzione adottata prevede, per ciascuna tipologia di flusso, una struttura a due livelli:
una **tabella di testata**, che rappresenta l'intera distinta ed è quindi collegata all'oggetto complesso di riferimento ed una **tabella di dettaglio** che rappresenta le singole "transazioni" interne.
Nella tabella di dettaglio, ogni record, oltre ad essere collegato alla testata corrispondente è ulteriormente collegato al record della transazione inviata.

Entrambe queste tabelle vengono parzialmente popolate già al momento dell'invio (attarverso la lettura del file inviato) in modo da rendere più agevole la riconciliazione (e quindi l'update dei record interessati).

Questa architettura riprende il classico modello **Master-Detail** ampiamente consolidato nella progettazione di sistemi informativi gestionali, e si dimostra particolarmente adatta a rappresentare la natura gerarchica dei flussi bancari, dove un'unica distinta può contenere decine o centinaia di transazioni differenti.

Dal punto di vista strategico, questa scelta consente di rispondere a due esigenze distinte ma complementari: avere una visione aggregata dello stato di ogni invio — la distinta nel suo complesso è stata accettata? ha generato errori? — e al contempo disporre del dettaglio granulare necessario per identificare eventuali anomalie a livello di singola transazione e intervenire in modo mirato.

Vi è però una terza funzione, meno immediata ma altrettanto rilevante: *la struttura a due livelli costituisce una forma di monitoraggio interno che il sistema aziendale costruisce autonomamente per compensare i limiti di visibilità offerti dalla controparte bancaria.*
Ritter, May e Rinderle-Ma segnalano esplicitamente come, nelle architetture di integrazione ibride, il monitoraggio dei flussi diventi più difficile proprio perché le interfacce esposte dai sistemi esterni possono essere limitate e non sempre consentono un'osservazione granulare dello stato delle operazioni [8, sez. 1.1].
Gli stessi autori osservano come il monitoraggio di scenari che attraversano più piattaforme — ad esempio tra ambienti on-premise e servizi cloud di terze parti, come nel caso della piattaforma bancaria — rimanga una sfida di ricerca ancora aperta, per la quale i pattern di osservabilità tradizionali offrono solo soluzioni parziali [8, sez. 7.1].

Costruire internamente al gestionale una rappresentazione strutturata di ciò che è stato inviato e di ciò che è stato ricevuto significa quindi non affidarsi passivamente alle informazioni restituite dalla banca, ma dotarsi di una base di conoscenza propria, interrogabile in qualunque momento e indipendente dalla disponibilità del sistema esterno.
In questo senso, il modello dati non è una semplice scelta implementativa, ma riflette una precisa visione del processo aziendale sottostante: la gestione dei pagamenti non è un'operazione monolitica, ma un flusso articolato che richiede visibilità ad ogni livello di granularità e che non può dipendere interamente dalla trasparenza della controparte per essere governato efficacemente.

4.2 La richiesta di file di ritorno

La ricezione degli esiti avviene tramite il servizio *Message* del gateway bancario, invocato periodicamente dall'operatore grazie al nuovo pulsante presente nella toolbar, il quale, in modo speculare al pulsante di invio, comunica con le API della piattaforma bancaria.

In risposta ad una chiamata GET, eventualmente arrichita da filtri, quali la data, il tipo di flusso o il codice SIA del cliente, il servizio restituisce un archivio compresso

contenente i file prodotti dalla banca relativi alle disposizioni precedentemente inviate.

A questo livello, i codici di risposta HTTP assumono un significato strettamente operativo, indipendente dal contenuto dei file: un codice **204** indica che non sono presenti esiti da analizzare — condizione attesa e non anomala, che segnala semplicemente che la banca non ha ancora prodotto esiti;
un codice **206** segnala che i file disponibili superano la soglia consentita in una singola risposta, suggerendo all'operatore di iterare le chiamate fino alla ricezione di un 200 o 204, ovvero la segnalazione di completamento del ciclo;
Infine, il classico codice **404** indica invece l'assenza di file in una situazione in cui ci si aspetterebbe di trovarne, configurando una condizione anomala che richiede verifica dei parametri e, se necessario, escalation operativa.

Il servizio a bancario, per facilitare e agevolare l'operatore, ha messo a disposizione il **meccanismo del cursore** il quale garantisce la continuità tra sessioni successive: il sistema mantiene traccia dell'ultimo messaggio scaricato per ciascuna combinazione di parametri di filtro, consentendo alle chiamate successive di riprendere esattamente dal punto in cui ci si era interrotti ed eliminando la necessità di gestire internamente la deduplicazione dei file ricevuti.

4.3 Il ciclo di vita delle disposizioni: testata e dettaglio

La struttura dati a due livelli descritta in precedenza non è soltanto un modello per organizzare le informazioni, ma la base concreta sulla quale è possibile la ricostruzione e la gestione del ciclo di vita di ogni distinta nella sua interezza.

Ciascun livello — testata e dettaglio — evolve in modo semi-indipendente, riflettendo perfettamente la natura gerarchica dell'intero processo: l'esito complessivo della distinta e l'esito delle singole transazioni che la compongono sono informazioni correlate ma differenti, da tracciare separatamente.

4.3.1 Il ciclo di vita della testata

Il ciclo di vita della testata inizia nel momento in cui il file viene generato e trasmesso all'hub bancario.
In questa fase, la tabella di testata viene popolata con le informazioni disponibili e recuperabili dalla lettura istantanea del file appena inviato, quali: identificativo della distinta, data di invio, SIA del mittente e ABI del ricevente.
In concomitanza all'operazione di salvataggio l'*operation result* viene impostato ad *In attesa di validazione*.
Da questo momento il sistema ha esaurito le proprie responsabilità dirette sull'operazione e si affida al ciclo di polling per ricevere le informazioni successive.

Come illustrato nel modello di transizione degli stati riportato in Figura 4.1, dallo stato di *In attesa di validazione* la testata può transitare verso uno solo di tre possibili

esiti, ciascuno definitivo: non esiste, in altre parole, un percorso che permetta ad una distinta/disposizione di tornare indietro o di essere corretta e reinviata sotto lo stesso identificativo.
Questo vincolo, che a primo impatto potrebbe sembrare rigido e limitante, è in realtà una scelta progettuale deliberata: preservare l'integrità dello storico significa che ogni distinta, una volta conclusa la propria elaborazione, rimane una testimonianza immutabile di ciò che è realmente accaduto, senza che correzioni successive possano alterarne la rappresentazione.

Nel caso **positivo**, ovvero nel caso in cui ogni singola transazione interna alla distinta sia correttamente andata a buon fine, l'*operation result* transita ad *Accepted*.
È importante sottolineare che questo esito non viene mai comunicato esplicitamente dalla banca: a differenza del totale o parziale rifiuto, che generano sempre un file di ritorno, il buon fine di una distinta si manifesta come assenza di segnalazioni.
È quindi lo stesso gestionale che si occupa di *inferire* l'esito positivo, verificando — tramite una funzione interna dedicata — che sia trascorso un intervallo di tempo superiore alla soglia di rischio insoluto definita dalla banca assuntrice, senza che nel frattempo sia stato ricevuto alcun esito negativo.
Solo a quel punto il sistema può "dare per scontato" il buon fine del pagamento e aggiornare l'*operation result* di conseguenza.
Si tratta di un meccanismo di conferma per silenzio, un pattern non infrequente nei protocolli di integrazione asincrona, ma che richiede un'attenzione particolare in fase di progettazione: l'assenza di un segnale negativo deve essere distinta, con certezza temporale, dalla semplice mancanza di elaborazione.

Nel caso **negativo**, la banca ha rifiutato l'intera distinta — tipicamente per problemi strutturali del file, quali errori di formattazione o parametri non conformi agli standard interbancari.
L'*operation result* viene impostato a *Rejected* e l'informazione viene resa disponibile agli operatori per le necessarie azioni correttive.

Il caso più frequente nella pratica operativa è tuttavia quello **parziale**: la distinta viene accettata nel suo complesso, ma alcune transazioni al suo interno risultano insolute.
In questo scenario l'*operation result* viene impostato a *Ritorno con Insoluti*, segnalando che è necessaria un'analisi granulare a livello di dettaglio per identificare le transazioni da rilavorare.

Proprio in virtù del vincolo di immutabilità descritto in precedenza, né una distinta *Rejected* né una distinta in *Ritorno con Insoluti* possono essere corrette e reinviate: la distinta originale, una volta raggiunto uno di questi due stati terminali, non viene più modificata.
Il rimedio operativo non avviene quindi a livello di testata, ma a livello di singola transazione, attraverso una sezione del gestionale dedicata — *Gestione Insoluti* — il cui funzionamento esula dall'ambito di analisi del presente lavoro, ma il cui scopo è semplicemente quello di "liberare" le transazioni fallite dal vincolo con la distinta di origine, rendendole disponibili per essere ricondotte in una nuova distinta e sottoposte nuova-

mente all'intero iter di invio.
In questo modo, lo storico della prima distinta resta intatto e l'errore viene risolto generando un nuovo ciclo di vita, anziché alterando quello già concluso.

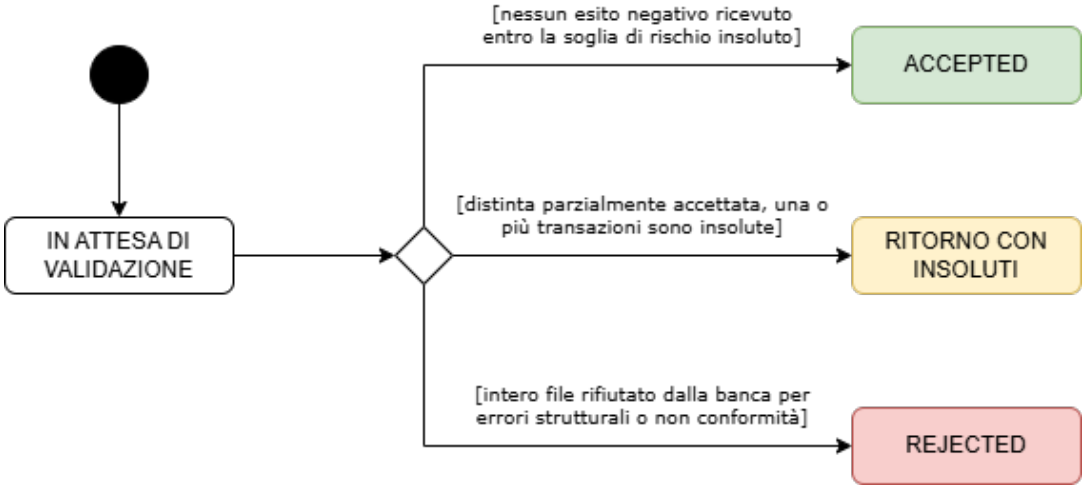

Figura 4.1: Diagramma degli stati del ciclo di vita della testata della disposizione. [13]

4.3.2 Il ciclo di vita del dettaglio

Proprio come la tabella di testata, anche quella relativa al dettaglio viene parzialmente popolata già al momento dell'invio. Per ciascuna transazione contenuta nella distinta viene infatti creato un record, con il numero dell'effetto e il riferimento alla transazione inviata già impostati, e con *response code* settato a *In attesa di ritorno*.
Questa scelta progettuale non è una semplice convenienza tecnica: significa che il sistema dispone, fin dal momento dell'invio, di una rappresentazione completa e minuziosa di tutto ciò che è stato trasmesso, indipendentemente da ciò che la banca deciderà — o non deciderà — di restituire in seguito.
È, in altre parole, la stessa logica di monitoraggio interno discussa nella prima sezione del capitolo, applicata questa volta al livello più fine di granularità disponibile.

Il ciclo di vita del dettaglio, tuttavia, non procede in modo uniforme per tutti i record. Quando la testata transita a *Ritorno con Insoluti*, il file restituito dalla banca contiene esclusivamente le transazioni che hanno generato un'eccezione: per ciascuna di esse, il sistema aggiorna il *response code* con il codice specifico fornito dalla banca, che ne identifica la motivazione del mancato incasso.
I record relativi alle transazioni assenti dal file — quelle, presumibilmente, andate a buon fine — non vengono invece toccati da alcun aggiornamento esplicito, e restano formalmente ancora in stato *In attesa di ritorno*.

È qui che si manifesta, a livello di singola transazione, la stessa logica di conferma per silenzio già descritta per la testata: un record che resta in *In attesa di ritorno* non può essere considerato pagato per il solo fatto di non essere stato segnalato come insoluto, poiché potrebbe semplicemente non essere ancora stato elaborato dalla banca.

Solo al superamento della soglia dei *giorni di rischio insoluto* il sistema può aggiornare il *response code* ad un esito di buon fine.
Fino a quel momento, ogni record privo di un esito esplicito rimane in uno stato di sospensione che non è né conferma né rifiuto.

Questa duplice applicazione del medesimo principio — a livello di distinta e a livello di singola transazione — evidenzia come l'asimmetria informativa imposta dal sistema bancario non sia un problema isolato, ma una caratteristica strutturale che permea l'intero processo di riconciliazione, a ogni livello di granularità.
La banca comunica le eccezioni; il silenzio, interpretato correttamente alla luce del tempo trascorso, diventa esso stesso un'informazione costitutiva del sistema.

Capitolo 5

Valutazione e conclusioni

Il percorso descritto in questo elaborato nasce da un problema che sembrava circoscritto: integrare il gestionale aziendale con il sistema bancario tramite le API del gateway.
Si è però rivelato un caso di studio rappresentativo di una sfida più ampia: costruire, sopra un'infrastruttura esterna che impone i propri vincoli e la propria opacità, un sistema interno capace di governare un intero processo senza dipendere passivamente dalla controparte.

I problemi affrontati in questa tesi — il polling asincrono, l'eterogeneità dei formati, la conferma per silenzio e l'asimmetria informativa tra i dati inviati e le risposte ricevute — non sono specifici dell'hub finanziario scelto.
Sono gli stessi problemi che si incontrano in qualsiasi integrazione in cui un sistema interno deve collaborare con un servizio esterno di terze parti, soprattutto se operante in un settore regolamentato come quello finanziario.

L'intermediario bancario utilizzato in questo lavoro è quindi la controparte concreta con cui ci si è confrontati, ma non è un limite del sistema costruito.
L'architettura a plugin è stata progettata per astrarre questa complessità e rimanere trasparente per il gestionale che la ospita: questo lascia aperta la possibilità di estendere in futuro l'integrazione ad altri hub finanziari, senza dover ridisegnare i principi su cui il sistema è costruito.
Il valore di questo lavoro sta proprio nel dimostrare che, anche partendo da un'interfaccia esterna vincolata e poco trasparente, è possibile costruire un livello applicativo che permette di governare l'integrazione, invece di limitarsi a subirla.

5.1 Dal frammento al sistema

Prima di questo intervento, il sistema aziendale non disponeva di alcuna rappresentazione strutturata del processo di pagamento: ogni transazione generava un file a sé, e l'intera gestione — dall'invio alla verifica dell'esito — era affidata interamente all'operatore, costretto a ricostruire manualmente, record per record, lo stato di ciascuna disposizione.

Oggi, grazie alla ristrutturazione introdotta, ogni distinta è un oggetto complesso, coerente e tracciabile lungo tutto il suo ciclo di vita.
Non si tratta esclusivamente di un incremento dell'efficienza operativa, bensì di un cam-

biamento radicale nel controllo informativo: l'ERP è ora in grado di determinare, in qualsiasi momento, lo stato logico e transazionale di ogni singolo pagamento.

L'impatto di questo salto di qualità architetturale è evidente: l'utente contabile non è più l'esecutore manuale di un processo frammentato, ma assume il ruolo di semplice supervisore di un sistema che gestisce autonomamente la quasi totalità dei flussi, intervenendo esclusivamente in caso di eccezioni o validazioni bloccanti.

I riscontri misurabili di questa transizione, raccolti ex-ante rispetto al rilascio in produzione della release 6.13 di SAM ERP 2.0, sono stati strutturati all'interno di un'analisi quantitativa dedicata, i cui risultati evidenziano il valore pratico percepito sia dai tecnici che dagli utenti finali.

5.2 Validazione preliminare tramite questionario percettivo

Il questionario [1], distribuito in forma anonima a un campione di circa 30 operatori aziendali, da consulenti applicativi ai tecnici di assistenza e specialisti d'area, è stato strutturato su scala lineare da 1 a 5 e diviso in due macro-aree: impatto operativo e soddisfazione attesa da un lato, percezione dei rischi e del change management dall'altro.

È importante sottolineare che i dati presentati in questa sezione si riferiscono a un campione di N=31 risposte raccolte fino al primo luglio 2026. La raccolta è proseguita successivamente e ha assunto una valenza che va oltre il perimetro strettamente accademico di questo lavoro: i risultati aggregati sono stati infatti acquisiti anche come strumento di analisi interna da parte di Centro Software.

5.2.1 Impatto operativo e soddisfazione

Le risposte raccolte sulla prima macro-area, relativa al risparmio di tempo, alla riduzione degli errori manuali e all'eliminazione dei task ripetitivi, mostrano un orientamento ampiamente positivo.

La totalità degli intervistati ha infatti attribuito un punteggio di 4 o 5 alla domanda relativa alla riduzione del rischio di errori materiali rispetto al processo manuale precedente — un consenso unanime che riflette la percezione diffusa di un miglioramento strutturale nella qualità dei dati trasmessi.

Sul fronte del risparmio di tempo, il 96.8% del campione ha valutato come significativo o estremamente significativo il beneficio introdotto dall'automazione dei flussi di invio, con oltre il 32% che ha espresso il punteggio massimo.

Anche la percezione di una situazione contabile e bancaria più aggiornata — e di conseguenza di decisioni operative più rapide — si conferma solida: la domanda corrispondente ha ottenuto una valutazione media di 4.39 su 5, con l'87.1% delle risposte concentrate sui valori 4 e 5.

5.2.2 Percezione dei rischi e change management

La seconda macro-area, riguardante i timori e le potenziali resistenze date dal nuovo modello operativo, ha fatto emergere un quadro nettamente più articolato.

In particolare, alla domanda relativa alla "mancanza di controllo diretto" derivante dalla delega completa delle comunicazioni bancarie al software (che elimina il controllo visivo tipico dell'upload manuale) il 58% degli intervistati ha attribuito un punteggio di 4 o 5, il che indica che la maggioranza del campione ritiene che la transizione richieda un'azione mirata, sia tecnica — comunicazione trasparente dei meccanismi di sicurezza e crittografia adottati — sia commerciale, per rassicurare l'utente finale sul reale livello di controllo garantito dal sistema.

Sul fronte del cambiamento procedurale legato all'introduzione degli oggetti complessi, il quadro è più sfumato: la distribuzione delle risposte si concentra sul valore centrale della scala (41.9% al punteggio 3), con il 29% che esprime una preoccupazione più marcata (punteggio 2) e il restante 29% che non ritiene il cambiamento particolarmente problematico (punteggi 4 e 5). La valutazione media suggerisce un timore moderato ma non trascurabile: il nuovo vincolo procedurale — che impone l'associazione delle transazioni a una disposizione prima della generazione del file anche per gli utenti del solo pacchetto base — è percepito come un cambiamento da accompagnare con adeguata formazione.

Sul piano dell'appetibilità commerciale del modulo come pacchetto aggiuntivo, il campione esprime invece un orientamento nettamente positivo: il 90.3% degli intervistati ha attribuito un punteggio di 4 o 5, indicando che il valore operativo percepito — automazione dei flussi, azzeramento dei tempi di upload, riduzione del rischio di errori — è ritenuto sufficiente a giustificare l'investimento commerciale richiesto al cliente.

5.2.3 Considerazioni di sintesi

Pur trattandosi di un riscontro preliminare, i risultati raccolti offrono un primo segnale incoraggiante, in particolare per quanto riguarda l'efficienza operativa attesa.
Le perplessità emerse nella seconda macro-area, seppur più contenute, indicano che la fiducia nei meccanismi di sicurezza sottostanti non può essere data per scontata, al contrario richiede un'azione di comunicazione e formazione mirata verso l'utente finale.
Una validazione più solida di questi riscontri preliminari potrà essere condotta solo a seguito dell'adozione effettiva del modulo sul campo, una volta disponibili i dati di utilizzo reale.

5.3 Sviluppi futuri e considerazioni finali

Il modello architetturale sviluppato lascia spazio a più linee di evoluzione e completamento:

- **Modulo di Gestione Insoluti:** A seguito della ricezione di transazioni con esito negativo, un possibile sviluppo futuro introdurrebbe un'azione guidata direttamente nell'interfaccia che non si limiterebbe al semplice scollegamento dalla distinta

originaria, ma proporrebbe all'operatore — tramite un apposito pulsante — la creazione automatica di una nuova distinta con le sole transazioni fallite già associate, pronta per la generazione del file e per un nuovo ciclo di invio.

- **Contabilizzazione automatica degli esiti:** Una volta che una transazione finanziaria raggiunge uno stato definitivo, il passo successivo consiste nel tradurre tale informazione in una scrittura contabile automatica nel modulo *Accounting* di SAM ERP 2.0, azzerando completamente l'inserimento manuale da parte dell'ufficio tesoreria.

- **Dashboard informativa direzionale:** Sfruttando la struttura a due livelli (testata e dettaglio) implementata per gli oggetti complessi, è possibile realizzare un'interfaccia di monitoraggio aggregata in tempo reale. Tale cruscotto permetterebbe di visualizzare metriche prestazionali immediate, quali il volume delle distinte in attesa di validazione, i tassi di insoluto e le tempistiche medie di elaborazione dei flussi.

Le linee di sviluppo qui descritte non intaccano la solidità del nucleo realizzato in questo lavoro, ma ne rappresentano il naturale completamento: punti che, per vincoli di tempo o di ambito, non è stato possibile affrontare nel periodo della presente release, ma che la struttura architetturale adottata rende possibile sviluppare senza dover rimettere in discussione le scelte progettuali di fondo.

In conclusione, questo progetto ha confermato che la digitalizzazione dei processi aziendali non è mai un fatto puramente tecnologico.
È un processo che richiede la capacità di leggere l'organizzazione, di capire come le persone lavorano e di progettare sistemi che migliorino quel lavoro senza stravolgerne le logiche consolidate.
La tecnologia è lo strumento; la comprensione del processo è la competenza che ne determina il valore.

Il percorso da tirocinante a collaboratrice, parallelo all'evoluzione del progetto stesso, ha reso questa esperienza qualcosa di più di un caso studio: è stata l'occasione di verificare sul campo che le scelte progettuali descritte in questa tesi non sono esercizi astratti, ma decisioni con conseguenze reali su persone reali che useranno questo software ogni giorno.

Riferimenti Bibliografici

[1] L. Gotti, *Questionario di Valutazione Impatto Operativo: Integrazione dell'hub bancario (SAM ERP 2.0)*, Questionario somministrato in forma anonima tramite Google Moduli, Dato primario raccolto dall'autrice nell'ambito del presente lavoro di tesi, 2026. indirizzo: `https://drive.google.com/file/d/18tf6ml-ujBc3VZZkrgUlSoWzdawgeIoM/view?usp=sharing`.

[2] Wikisoftware, *BICTA - Gateway per l'integrazione dei flussi bancari e servizi Fintech*, `https://wikisoftware.it/bicta-cosa-fa/`.

[3] T. Federici, "The impact of ERP systems in Italian SMEs," *Proceedings of ITAIS*, 2006.

[4] E. Gamma, R. Helm, R. Johnson e J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[5] Y. Touil, "Architectural Evolution of Microsoft ERP Systems: From Monolithic Dynamics NAV to Event-Driven Business Central Extensions," Microsoft Dynamics 365 Business Central Solutions, Architectural Study, 2026. indirizzo: `nav-to-business-central-architecture-study.pdf`.

[6] B. Meyer, *Object-Oriented Software Construction*. Prentice Hall, 1988.

[7] G. Hohpe e B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* (Addison-Wesley Signature Series). Addison-Wesley Professional, 2003, ISBN: 9780321200686.

[8] D. Ritter, N. May e S. Rinderle-Ma, "Patterns for Emerging Application Integration Scenarios: A Survey," *Information Systems*, vol. 67, pp. 36–57, 2017.

Indice delle Fonti Iconografiche

[9] Linda Gotti. “Diagramma di stato del ciclo di vita di una disposizione/distinta.” Modellazione originale dell’autrice.

[10] Linda Gotti. “Diagramma delle Attività del processo di RedirectEvent Pre-Processing.” Modellazione originale dell’autrice.

[11] Linda Gotti. “Schema riassuntivo ad alto livello di Astrazione del ruolo del gateway bancario.” Modellazione originale dell’autrice.

[12] Linda Gotti. “Diagramma di sequenza dal momento dell’autenticazione all’invio del file.” Modellazione originale dell’autrice basata sulle specifiche tecniche fornite dal servizio intermediario.

[13] Linda Gotti. “Diagramma di stato del ciclo di una distinta dall’invio alla ricezione.” Modellazione originale dell’autrice basata sulle specifiche tecniche fornite dal servizio intermediario.