

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

SCUOLA DI INGEGNERIA E ARCHITETTURA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

PROGETTAZIONE E IMPLEMENTAZIONE DI
UNA TRANSAZIONE IN RADIOFREQUENZA PER
ATTIVITÀ DI INVENTARIO FISICO SU SISTEMA
SAP-EWM

Elaborata nel corso di: Project Management

Tesi di laurea di:
MANFREDI CHRISTIAN

Relatore:
Chiar.mo Prof. MARCO
ANTONIO BOSCHETTI

Correlatore:
Dott. ALESSANDRO ZAVINO

ANNO ACCADEMICO 2017-2018 (SESSIONE I)

PAROLE CHIAVE

SAP

Extended Warehouse Management

ABAP

Transazione in Radiofrequenza

Project Management

Sommario

Negli ultimi decenni il mercato dei prodotti software destinati alla gestione aziendale ha subito una notevole espansione grazie allo sviluppo di tecnologie informatiche innovative indirizzate alla risoluzione dei problemi legati a questo determinato ambito applicativo. In tal senso, occupano un posto di primaria importanza i sistemi software ERP, Enterprise Resource Planning, che nascono dall'esigenza delle aziende di porre rimedio alla frammentazione del mercato di questi prodotti integrando all'interno di un unico sistema software tutti i processi gestionali.

In questo contesto operativo si inserisce il presente lavoro di tesi, nato dall'esigenza di una primaria azienda internazionale, produttrice di elettrodomestici, di sviluppare ed integrare un processo ad hoc, all'interno del software ERP SAP, per le attività di inventario fisico di uno degli impianti di stoccaggio e distribuzione di parti di ricambio a livello europeo.

La soluzione progettata ed implementata, oggetto di questo elaborato, ha l'obiettivo di permettere agli operatori di magazzino di portare a termine le operazioni di conteggio delle rimanenze inventariali con maggiore efficienza, riducendo gli errori in fase di immissione dei dati ed integrando tra le funzionalità del prodotto un meccanismo per la verifica automatica dei documenti di inventario.

Ringraziamenti

Innanzitutto vorrei esprimere la mia gratitudine al mio relatore, il professor Marco Antonio Boschetti, per avermi dato la possibilità di vivere l'esperienza di tirocinio aziendale che ha portato alla stesura del presente elaborato di tesi. Devo inoltre ringraziarlo per il continuo supporto e la fiducia mostrata nei miei confronti.

Un sentito ringraziamento a Bruno Bartoletti, Alessandro Zavino e agli altri dipendenti di Engineering Ingegneria Informatica s.p.a. presso la sede operativa di Rimini, per avermi concesso l'opportunità di conoscere la piattaforma software ERP SAP ed avermi guidato nel mio percorso progettuale.

Infine, ultimo ma non per importanza, un ringraziamento speciale va ai miei genitori, a tutti i miei familiari e ai colleghi, per avermi sostenuto ed incoraggiato nella mia carriera universitaria.

Indice

Sommario	v
Ringraziamenti	vii
Introduzione	xiii
1 Dominio applicativo	1
1.1 Sistema ERP SAP	1
1.1.1 Analisi dell'architettura di sistema	4
1.1.2 Ambiente di sviluppo	8
1.2 SAP Extended Warehouse Management	11
1.2.1 Comunicazione con il sistema ERP	12
1.3 Attività di inventario fisico	13
1.3.1 Inventory Management and Physical Inventory	14
2 Tecnologie di riferimento	17
2.1 Il linguaggio ABAP	17
2.1.1 Caratteristiche del linguaggio ABAP	18
2.1.2 Esecuzione di programmi ABAP in ambiente SAP	25
2.2 Framework per transazioni in radiofrequenza	26
3 Caso di studio	29
3.1 Analisi del problema	30
3.2 Obiettivo del progetto	30
3.3 Analisi dei requisiti della soluzione	30
3.4 Criteri di successo	31
3.5 Assunzioni e rischi	32

4	Implementazione della transazione RF personalizzata	33
4.1	Operazioni preliminari	33
4.2	Customizing	35
4.3	Creazione della transazione ZIVCOU	41
4.3.1	Creazione delle schermate personalizzate	43
4.4	Implementazione delle modifiche funzionali	47
4.4.1	Compilazione automatica dei campi	49
4.4.2	Implementazione della funzionalità di conferma auto- matica dei documenti di inventario	53
4.4.3	Esecuzione della transazione logica	61
5	Realizzazione del Report di revisione dei documenti di in- ventario	67
5.1	Definizione dell'interfaccia	68
5.2	Implementazione delle funzionalità	72
	Conclusioni	77
	Appendice A	79
	Bibliografia	113

Elenco delle figure

1.1	Evoluzione dell'architettura della piattaforma SAP	3
1.2	SAP NetWeaver	4
1.3	Esempio di Change Request	7
4.1	Creazione Change Request di tipo Workbench	34
4.2	Creazione Package di sviluppo	35
4.3	Ramo di Customizing per gestire l'interfaccia utente	36
4.4	Creazione di un Display Profile in copia	38
4.5	Presentation Profile e Personalization Profile	40
4.6	Dialog di copia delle dipendenze della transazione di partenza	41
4.7	Finestra di mappatura delle sotto-schermate	42
4.8	Schermata 9999 analizzata con lo strumento Screen Painter .	44
4.9	Diagramma delle modifiche apportate alla schermata	45
4.10	Schermata 9999 definitiva	46
4.11	Flusso PBO-PAI in transazione RF	48
4.12	Definizione del flusso di una transazione RF	49
4.13	Tabella custom ZSPC_PARM	54
4.14	Tuple inserite per il posting automatico dei documenti	55
4.15	Struttura della tabella ZSPC_PHY_DOC	56
4.16	Percorso menu per la transazione ZIVCOU	62
4.17	Scansione della Bin da verificare	63
4.18	Scansione di una HU all'interno dell'ubicazione	64
4.19	Processo di conteggio dei prodotti presenti nel contenitore .	65
5.1	Dialog di configurazione del programma	69
5.2	Finestra di creazione di una nuova schermata	70
5.3	Tabella ZSPC_PHY_DOC visualizzata mediante ALV Grid	72
5.4	Popup di conferma alla creazione di un nuovo documento . .	74

Introduzione

Negli ultimi decenni il mercato dei prodotti software destinati alla gestione aziendale ha subito una notevole espansione grazie allo sviluppo di tecnologie informatiche innovative indirizzate alla risoluzione dei problemi legati a questo determinato ambito applicativo. La gestione delle imprese, a partire dalle più modeste fino ad arrivare alle multinazionali, è stata riformata e strutturata anche grazie all'ausilio di piattaforme software in grado di monitorare con precisione ogni aspetto delle attività aziendali in modo da garantire un aumento significativo dell'efficienza e dell'efficacia dei processi gestionali.

In particolare, occupano un posto di primaria importanza i sistemi software denominati ERP, Enterprise Resource Planning. Questi sistemi informativi nascono dall'esigenza delle aziende di porre rimedio alla frammentazione del mercato dei prodotti informatici dedicati alla gestione aziendale per integrare all'interno di un unico sistema software tutti i processi gestionali. Nello specifico, questi prodotti permettono di ottenere una migliore circolazione delle informazioni in ambito aziendale grazie al soddisfacimento di tre fondamentali requisiti:

- **Unicità dell'informazione:** garantisce che un dato sia valorizzato in egual modo all'interno di ogni processo gestionale grazie all'adozione di una base di dati condivisa capace di garantire (1) la corretta sincronizzazione e l'affidabilità delle informazioni, (2) l'assenza di ridondanza per ridurre le tempistiche delle attività di mantenimento dei dati e (3) la tracciabilità degli aggiornamenti per una migliore sicurezza.
- **Modularità del sistema:** grazie alla quale è possibile gestire con maggiore efficienza estensioni o aggiornamenti del software.

- **Prescrittività:** i sistemi ERP non vengono costruiti a partire dai processi implementati in un'azienda, ma definiscono a priori le logiche operative da utilizzare sulla base delle migliori pratiche disponibili nell'ambito operativo, focalizzando l'attenzione sulla parametrizzazione del software sulla base alle esigenze della singola impresa.

La nascita di questa nuova tipologia di sistemi informativi ha permesso l'evoluzione del mercato verso la produzione di piattaforme software modulari capaci di coprire la totalità dei processi aziendali in modo più o meno approfondito e di garantire l'aderenza agli standard di settore più utilizzati. A titolo non esaustivo, si elencano alcune delle attività gestite dai software ERP:

- Contabilità e gestione del credito.
- Assistenza clienti.
- Gestione materiali.
- Gestione magazzino e trasporti.
- Gestione commerciale (distribuzione e vendite).
- Pianificazione della produzione.
- Controllo della qualità.
- Gestione delle risorse umane.

Uno dei maggiori esponenti a livello mondiale di questa categoria di applicativi è **SAP** - Systeme, Anwendungen, Produkte in der Datenverarbeitung - sviluppato dalla società SAP-SE con sede a Walldorf. Il sistema nasce negli anni '70 come software per la contabilità aziendale e, grazie ad uno sviluppo puntuale e costante negli anni successivi, rappresenta oggi il prodotto ERP leader per la gestione di imprese di grandi dimensioni nonché uno dei più articolati con i suoi innumerevoli moduli funzionali.

Tra i processi aziendali informatizzati da SAP, uno dei più importanti è rappresentato dalle attività di **inventario fisico**. Queste sono utilizzate all'interno di una struttura di stoccaggio, quale ad esempio un magazzino per la distribuzione o un deposito di stoccaggio legato ad un impianto di

produzione, per verificare la presenza fisica dei beni sia ai fini della contabilità fiscale sia per mantenere il sistema informativo allineato con le effettive disponibilità. Inoltre, da un punto di vista normativo, è obbligatorio effettuare questa operazione almeno con cadenza annuale in vista della chiusura di bilancio, per scopi di rendicontazione delle rimanenze inventariali.

Nel tempo sono nate numerose realtà aziendali specializzate nella consulenza e nella personalizzazione della piattaforma ERP sviluppata da SAP in base alle esigenze specifiche del cliente. Una di queste è rappresentata da **Engineering Ingegneria Informatica s.p.a.** che, nell'ambito operativo di una delle sue divisioni aziendali, si occupa dell'implementazione di questo software, e di alcuni suoi moduli, su richiesta di partner di grande rilevanza nel mercato. In particolare, la società ha partecipato alla realizzazione di numerosi progetti internazionali di grandi dimensioni volti all'automatizzazione di processi di stoccaggio e trasporto di beni.

Obiettivi

All'interno di questo contesto si inserisce il presente lavoro di tesi, nato dall'esigenza da parte di un grande impianto di stoccaggio e distribuzione di parti ricambio a livello europeo, appartenente ad una primaria azienda internazionale produttrice di elettrodomestici, di implementare un processo ad hoc per le proprie attività di inventario fisico. Gli operatori di magazzino preposti eseguono queste attività seguendo le direttive comunicategli dal sistema centrale utilizzando un palmare personale collegato ad un canale per le comunicazioni in radiofrequenza (RF). Il terminale, mediante l'utilizzo degli strumenti di input integrati, permette all'utente di inserire a sistema i risultati delle operazioni di conteggio in modo da aggiornare lo stato corrente delle rimanenze dell'impianto di stoccaggio.

Gli obiettivi che ci si è posti per questo progetto sono stati:

- Progettazione ed implementazione di una transazione in radiofrequenza personalizzata per l'esecuzione delle attività di inventario fisico in modo automatizzato, secondo le direttive imposte dalla società esterna incaricata della gestione finanziaria dell'impianto.
- Personalizzazione del sistema informativo con l'obiettivo di memorizzare dati aggiuntivi relativi alle operazioni di magazzino svolte dai dipendenti.

- Sviluppo di un report personalizzato per la gestione dei documenti di inventario segnalati in modo automatico dall'algoritmo di verifica delle differenze tra i dati presenti nel sistema informativo e quelli raccolti fisicamente all'interno dell'impianto. Da questo applicativo sarà possibile commissionare nuovi conteggi nel caso in cui questo risultasse necessario.

Si vuole evidenziare come tutti gli obiettivi prefissati sono stati conseguiti durante l'attività di tirocinio extra-curricolare in azienda che ha portato all'elaborazione del presente lavoro. In particolare, come mostrato in dettaglio nei capitoli successivi, è stata realizzata una nuova transazione in radiofrequenza, utilizzabile dagli operatori di magazzino per effettuare l'inventario fisico; il sistema informativo è stato arricchito mediante la rendicontazione di nuovi dati relativi a quest'attività; è stato sviluppato un applicativo per la visualizzazione e la gestione dei documenti di inventario bloccati dal sistema automatizzato.

Struttura della tesi

Il primo capitolo presenta una panoramica della piattaforma software SAP-ERP, sviluppata e distribuita da SAP SE, evidenziandone i punti di forza e le principali componenti. Successivamente, viene descritto il modulo della piattaforma Extended Warehouse Management (EWM) dedicato alla gestione delle principali attività di magazzino come prelievo e stoccaggio di materiali, organizzazione funzionale degli ambienti, gestione della ricezione e dell'invio di prodotti e inventario fisico, solo per citarne alcune. Infine, si analizza il dominio applicativo in cui si inserisce il lavoro descritto in questa tesi, le attività inventariali e la gestione fiscale dei prodotti stoccati, oltre ai processi presenti all'interno della piattaforma SAP-EWM per la gestione di queste attività.

Il secondo capitolo offre un'analisi delle tecnologie utilizzate nella realizzazione della soluzione implementata quali il linguaggio di programmazione ABAP, utilizzato per la programmazione all'interno della piattaforma SAP, ed il framework di lavoro per la realizzazione e l'utilizzo delle transazioni in radiofrequenza all'interno del magazzino.

Il terzo capitolo si occupa della formalizzazione del contesto applicativo all'interno del quale si inserisce la soluzione proposta, analizzando i requisiti

e gli obiettivi che hanno portato alla sua progettazione ed implementazione. Questi sono stati riportati, per una migliore comprensione, all'interno di un documento stilato secondo le buone pratiche di *Project Management*.

Nel terzo capitolo si descrive in dettaglio l'implementazione della transazione in radiofrequenza realizzata, approfondendo anche l'operazione di personalizzazione dell'ambiente RF con cui interagiscono gli operatori di magazzino.

Il quinto capitolo, invece, si concentra sulla creazione della schermata di report personalizzata utile alla revisione dei documenti di inventario che non hanno superato la verifica automatica in fase di salvataggio.

Infine, nel capitolo conclusivo, si analizzano le difficoltà incontrate durante la realizzazione e si evidenziano alcune caratteristiche del prodotto finale.

Capitolo 1

Dominio applicativo

Questo capitolo vuole offrire una panoramica di ampio respiro, anche se sufficientemente dettagliata, sul dominio applicativo del progetto sviluppato. Lo scopo è quello di permettere al lettore di acquisire le nozioni necessarie ad una lettura consapevole degli argomenti trattati nei capitoli successivi, in particolare le caratteristiche della piattaforma ERP di SAP ed i moduli inerenti al lavoro svolto.

Il capitolo consta di tre sezioni, così suddivise:

- La prima rappresenta un'analisi del sistema SAP, dell'evoluzione della sua architettura negli anni e degli strumenti di sviluppo messi a disposizione.
- Successivamente si mostrano le principali caratteristiche del modulo funzionale SAP Extended Warehouse Management, dedicato alla gestione dei processi di magazzino, soffermandosi anche sui meccanismi di comunicazione tra questo ed il sistema ERP.
- Infine si vuole offrire lo stato dell'arte dei processi di inventario fisico nel mondo aziendale ed, in particolare, come questi vengo gestiti all'interno di SAP.

1.1 Sistema ERP SAP

Uno dei prodotti ERP di maggior successo è la piattaforma sviluppata da SAP-SE, azienda fondata nel 1972 con sede a Walldorf, in Germania. L'a-

cronimo SAP si esplicita con "Systeme, Anwendungen, Produkte in der Datenverarbeitung" e può essere tradotto in Italiano come "Sistemi, Applicazioni e Prodotti nell'elaborazione di dati".

Il sistema nasce agli inizi degli anni '70 come software per la gestione della contabilità, basato sull'utilizzo di schede perforate e destinato all'installazione su Mainframe DOS di IBM. Questa prima versione venne utilizzata come base per la creazione della prima versione della suite di software per la gestione aziendale denominata R/1. Dopo circa un decennio l'architettura della piattaforma venne rivoluzionata per rispondere all'andamento del mercato, con l'adozione sempre maggiore di mainframe basati su sistemi operativi UNIX. Invece di costruire un sistema operativo proprietario capace di gestire la piattaforma hardware di riferimento, l'azienda decise di rendere i propri prodotti compatibili con l'esecuzione su sistemi UNIX ed adottò un'architettura a due livelli: i programmi (Application Layer) ed il database erano ancora eseguiti da un mainframe comune mentre gli operatori potevano interagire con il sistema mediante l'utilizzo di terminali basati su un'interfaccia a riga di comando (Presentation Layer). Il sistema venne denominato SAP R/2 ed offriva un ambiente di runtime, chiamato SAP Basis, all'interno del quale venivano eseguiti i programmi scritti mediante il linguaggio proprietario ABAP. Questo linguaggio permetteva la creazione di applicazioni in grado di interagire con i dati memorizzati all'interno del database e generare dei report statici.

L'avvento della diffusione dei PC e delle interfacce grafiche portò alla successiva evoluzione dell'architettura della piattaforma. Il sistema adottò una distribuzione su 3 livelli, delegando ai sistemi client la gestione dell'interfaccia grafica che, in prima istanza, veniva elaborata dal mainframe e successivamente presentata agli utenti sul loro dispositivo. Questa nuova architettura, chiamata R/3 in continuità con le precedenti, determinò quindi la suddivisione della piattaforma in client, server e database, come si può vedere in figura 1.1. La nuova versione del sistema informativo poteva essere considerata a tutti gli effetti un software ERP grazie agli oltre 40 business modules presenti, coprendo la maggior parte dei processi operativi di un'azienda. Il cuore del sistema era ancora SAP Basis, scritto mediante l'utilizzo del linguaggio C e capace di eseguire applicazioni ABAP/4. Inoltre, ogni elemento del software era ora memorizzato anch'esso all'interno del database, comprendendo sia le applicazioni che l'ambiente di sviluppo stesso.

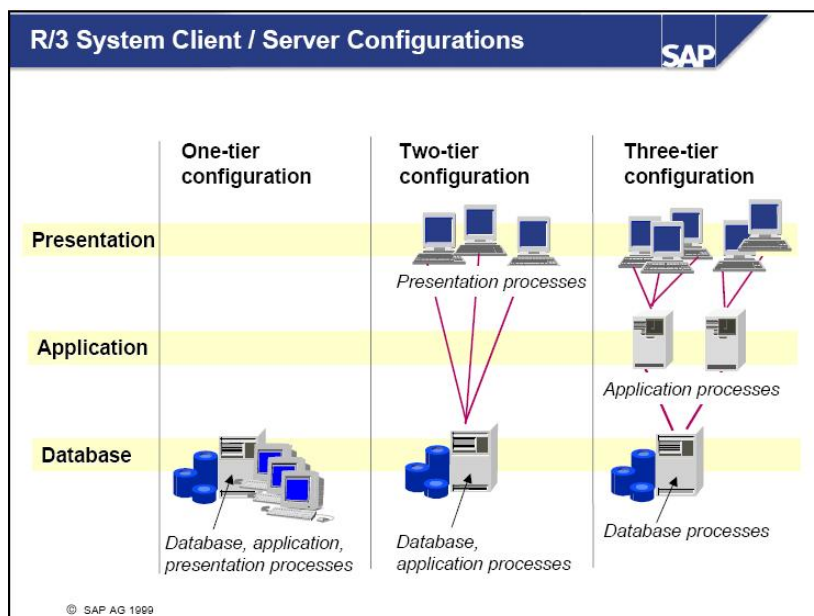


Figura 1.1: Evoluzione dell'architettura della piattaforma SAP

La più recente evoluzione del sistema SAP ERP avvenne nei primi anni del 2000 in risposta alla diffusione delle tecnologie Internet. Questa nuova versione mantiene la piena compatibilità con l'architettura SAP R/3 ed aggiunge il supporto all'accesso al sistema tramite browser Internet e sistemi basati su tecnologia Java grazie ad un Web server proprietario in grado di offrire un'interfaccia di comunicazione comune per lo scambio di informazioni con sistemi R/2, R/3 e di terze parti. La nuova piattaforma venne denominata SAP NetWeaver e il nuovo componente centrale del sistema, in sostituzione del precedente SAP Basis, venne chiamato SAP NetWeaver Application Server. Il linguaggio di programmazione proprietario ABAP fu profondamente evoluto, abbracciando il paradigma ad oggetti come suggerisce la nomenclatura ABAP Objects. In figura 1.2 viene mostrata una panoramica dell'architettura SAP NetWeaver e dei suoi componenti. Si può notare come ora l'ambiente di runtime permette l'esecuzione di programmi sia scritti in ABAP che in Java. Inoltre, a differenza delle prime versioni della piattaforma è ora possibile creare (1) report statici, mediante i quali visualizzare e modificare i dati prelevati dal database, (2) applicazioni Dynpro, caratterizzate da un flusso informativo su più schermate interattive -

dette dialog - e (3) Web Dynpro, accessibili mediante browser web in quanto codificati con il linguaggio di mark-up HTML5.

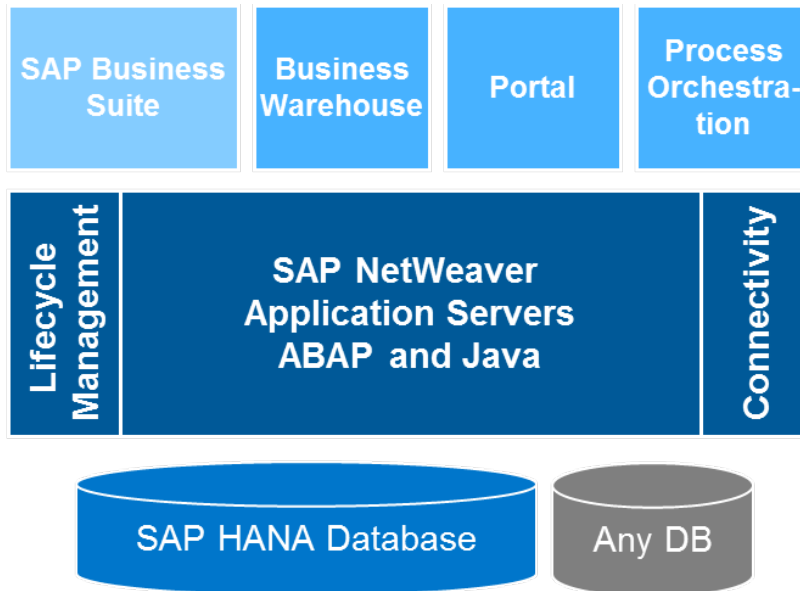


Figura 1.2: SAP NetWeaver

1.1.1 Analisi dell'architettura di sistema

In questa sezione dell'elaborato si vuole analizzare in dettaglio l'architettura tecnica di un tipico sistema SAP ERP in configurazione a tre livelli: *Presentation layer*, *Application layer* e *Database layer*, come mostrato in figura 1.1. La comunicazione, come solitamente avviene nelle architetture stratificate, è permessa solo con i livelli immediatamente superiori o inferiori in modo da garantire che il flusso di dati all'interno dell'applicativo segua una direzione precisa.

- **Presentation layer:** rappresenta un qualsiasi dispositivo di Input/Output utilizzato da un operatore per interagire con il sistema; permette sia di ricevere informazioni che di impartire comandi ed avviare processi. Sono supportati numerosi dispositivi tra cui, ma non solo, l'interfaccia **SAP GUI** su terminale compatibile, browser

e terminali in radiofrequenza. Può scambiare dati esclusivamente con l'*Application layer* sottostante.

- **Application layer:** rappresenta l'unità centrale di elaborazione dei dati e comprende la logica di business dell'intera piattaforma. Può essere formata da una o più istanze del sistema di processamento delle informazioni e comunica sia con il *Presentation layer* che con il sottostante *Database layer*.
- **Database layer:** la gestione della memorizzazione permanente dei dati viene separata dall'elaborazione degli stessi sia logicamente, ponendola su un diverso livello architetturale, sia fisicamente, in quanto le istanze sono eseguite su server separati per ragioni di performance e di sicurezza.

Oltre alla distinzione fra questi diversi livelli dell'architettura tecnica - suddivisione introdotta, si ricorda, a partire dalla versione R/3 della piattaforma SAP - si mostra di seguito anche la più comune configurazione di sistemi multipli adottata dagli utenti di SAP ERP ed i motivi per cui ciò avviene. Tipicamente, infatti, in un'azienda che adopera la piattaforma, in misura più o meno ampia, per la gestione aziendale si possono individuare 3 sistemi: il *Sistema di Sviluppo*, il *Sistema di Testing* ed, infine, il *Sistema di Produzione*. E' possibile che, per esigenze di sviluppo o di manutenzione, vengano utilizzati altri sistemi come, ad esempio, un sistema di training o un sistema di collaudo/consolidamento.

La ragione per cui si adotta questa strategia è relativamente semplice: separare nettamente gli ambienti dei vari sistemi permette di sviluppare e testare le correzioni o le nuove funzionalità assicurandosi che il sistema in uso dagli operatori, quello di produzione, non venga modificato prima che i dovuti test abbiano dato esito positivo. L'unico punto di contatto tra i differenti sistemi è rappresentato da un ulteriore componente, il **Sistema di Trasporto**, il cui compito è quello di trasferire le modifiche effettuate su un sistema al sistema successivo mediante l'utilizzo di pacchetti chiamati **Change Request**, analizzati in dettaglio in una sezione successiva di questo capitolo. Un altro motivo per cui viene effettuata questa separazione tra i diversi sistemi è la quantità di dati che caratterizza l'ambiente di produzione, utilizzato nelle operazioni di business dell'azienda. Spesso la mole di informazioni a sistema risulta troppo grande per l'utilizzo durante lo sviluppo e, per questa ragione, si decide di configurare un ambiente contenente

solo un sottoinsieme dei dati reali. Oppure, dato che molto frequentemente lo sviluppo di nuove funzionalità viene commissionato a compagnie esterne, l'azienda decide di adottare questa soluzione per ragioni di sicurezza, nascondendo i dati reali poichè considerati sensibili.

Ognuno dei sistemi menzionati è strutturato, solitamente, in un proprio Application Server ed utilizza un Database server separato, mantenendo la totale indipendenza di un ambiente da un altro. Da notare, inoltre, che all'interno di una installazione multi-sistema sono presenti due insiemi di parametri di configurazione:

- **Workbench settings:** impostazioni comuni a tutti i sistemi tra cui librerie standard SAP, function module - analizzati in seguito - personalizzati e tabelle.
- **Customizing:** variabili di funzionamento che caratterizzano un singolo sistema e ne determinano i meccanismi di funzionamento. Queste impostazioni vengono modificate mediante la transazione ABAP *SPRO*.

Sistema di trasporto degli aggiornamenti

La separazione in diversi sistemi di una comune configurazione rende necessario un meccanismo di trasporto delle modifiche da un sistema all'altro, assicurandosi che venga mantenuta l'integrità sia dei dati che dei processi utilizzati dal sistema di destinazione. Il meccanismo implementato all'interno della piattaforma SAP viene chiamato, come già accennato in precedenza, **Sistema di Trasporto** e raggruppa tutti gli strumenti necessari all'interno di un unico applicativo che effettua numerosi controlli in fase di trasferimento per garantire che ogni aggiornamento sia circoscritto ad un insieme finito di variazioni e che sia possibile, successivamente, ripristinare lo stato precedente del sistema di destinazione.

Affinchè dei cambiamenti possano essere trasportati in un altro sistema è obbligatorio creare un oggetto chiamato *Change Request* (CR) o *Transport Request*. Questo è un contenitore che permette di assegnare un identificatore univoco ad un insieme di modifiche in modo che l'intero processo di aggiornamento sia tracciabile. Un oggetto CR è composto di una o più unità minime trasportabili dette *Change Task* (CT), a cui è assegnato un unico utente che se ne assume la responsabilità. Ogni CT assicura che gli oggetti

in esso contenuti non vengano modificati da più utenti in task diversi. I contenitori CR, invece, vengono creati e rilasciati da un responsabile designato della gestione di tutti i task che li costituiscono. Sia CR che CT sono caratterizzati da uno stato di avanzamento: un CR può essere rilasciato solo quando tutti i CT che contiene vengono autorizzati alla distribuzione. Ogni CR può essere categorizzato, e di conseguenza variano anche le modalità di elaborazione, in base al tipo di modifiche che contiene. Nel caso in cui le modifiche riguardino i componenti appartenenti alle impostazioni Workbench, queste vengono definite *Workbench Request* ed i loro effetti si ripercuoteranno sul tutti i sistemi; se le modifiche sono invece circoscritte a dati propri di un singolo sistema vengono denominate *Customizing Request*. In figura 1.3 viene mostrata un Change Request di esempio, all'interno interfaccia SAP GUI.

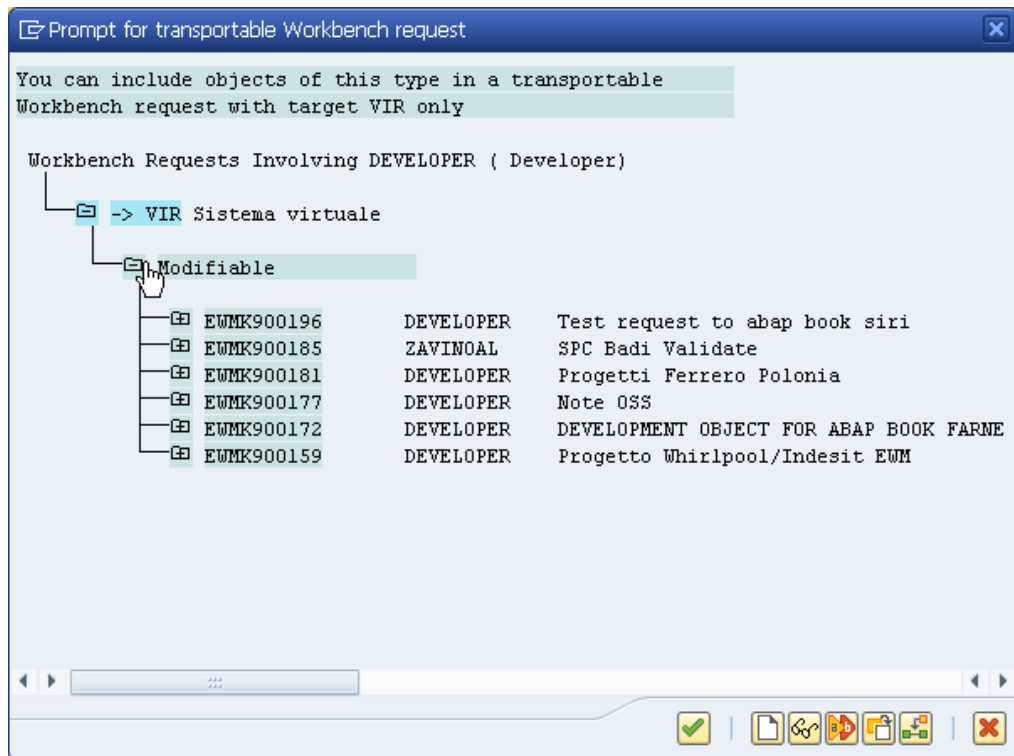


Figura 1.3: Esempio di Change Request

1.1.2 Ambiente di sviluppo

Tutti i membri di un team di progetto SAP operano, tipicamente, su di un singolo sistema, collaborando nella realizzazione degli obiettivi programmati. Questo ha pesantemente condizionato, in fase di sviluppo, il modo in cui SAP è stato strutturato ed implementato, sia in termini di strumenti a disposizione degli sviluppatori sia in termini di processi di elaborazione. Il sistema, infatti, raggruppa tutti gli applicativi necessari alla gestione di un progetto all'interno di un ambiente unico, **ABAP Workbench**, accessibile mediante interfaccia SAP GUI.

Prima di analizzare in dettaglio alcuni degli strumenti maggiormente utilizzati nello sviluppo di programmi in ambiente SAP, è necessario specificare quali sono le tipologie di applicativi utilizzati dalla piattaforma per le funzionalità destinate sia agli utenti finali che agli sviluppatori. SAP utilizza principalmente tre categorie di programmi:

- **Report:** come suggerisce il nome, rappresentano programmi con funzionalità limitate che hanno l'obiettivo di recuperare informazioni dal database secondo criteri specificati dall'utente e di mostrarli a schermo, mediante l'interfaccia grafica, solitamente sotto forma di una lista interattiva. I dati possono anche essere elaborati prima di essere visualizzati, ad esempio ricavando dinamicamente informazioni dipendenti da più fonti, ed il programma può offrire la possibilità di aggiungerne, modificarne o eliminarne una parte. Quando un utente esegue un report viene solitamente interrogato, mediante una schermata di selezione, per stabilire i criteri di ricerca e selezione dei dati di interesse; una volta comunicati i parametri i dati vengono visualizzati senza che l'utente possa intervenire ulteriormente nel processo di esecuzione. I programmi di questo tipo, inoltre, non sempre hanno la finalità di generare una lista di informazioni ma, spesso, si limitano ad implementare funzionalità di *business logic* senza generare un output.
- **Dynpro:** rappresentano programmi dinamici, formati da una o più schermate, che permettono all'utente di intervenire attivamente durante l'esecuzione. Le schermate che vanno a comporre l'applicazione vengono chiamate **Dialog** e sono collegate tra loro mediante azioni eseguibili dagli utenti; sono questi ultimi, infatti, a determinare

il flusso del programma interagendo con pulsanti e campi di input presentati a schermo. Gli applicativi Dynpro sono, quindi, maggiormente adatti alla creazione di interfacce grafiche complesse necessarie ad implementare funzionalità di elaborazione dei dati più sofisticate.

- **Web Dynpro:** sono applicazioni ABAP basate sull'utilizzo di tecnologie Web per la creazione di interfacce compatibili con il linguaggio HTML. Rispetto ai classici Dynpro non necessitano dell'installazione di SAP GUI sul sistema utilizzato dall'utente e sono orientati ad uno stile di programmazione dichiarativo mettendo a disposizione numerosi elementi grafici predefiniti liberamente utilizzabili per la costruzione di interfacce grafiche complesse e funzionali, limitando al contempo lo sforzo minimo di programmazione necessario. Inoltre, il framework messo a disposizione degli sviluppatori per la creazione di applicazioni di questo tipo implementa il pattern MVC, Model-View-Controller, maggiormente adatto alla gestione di programmi complessi e di dimensioni importanti.

E' importante notare come sia applicazioni report che Dynpro vengono eseguite all'interno dell'ambiente **ABAP Runtime Environment**, analizzato nel capitolo successivo, e fanno uso dell'interfaccia grafica SAP GUI installata sul sistema in uso. I programmi Web Dynpro, invece, nascono con l'obiettivo di essere eseguiti mediante un qualsiasi browser compatibile con lo standard Web HTML5, permettendone l'elaborazione sulla quasi totalità dei sistemi utilizzati senza richiedere ulteriori strumenti.

Transazioni SAP e strumenti di sviluppo

Numerosi applicativi messi a disposizione da SAP all'interno del sistema, siano essi destinati all'uso da parte degli utenti per l'esecuzione di processi aziendali o rivolti agli sviluppatori di programmi ABAP, vengono denominati **Transazioni**. Una transazione è un modulo software con specifiche funzionalità reso accessibile, in qualsiasi momento, dall'interfaccia SAP GUI mediante un identificativo detto *Transaction Code*. Ad oggi, le transazioni disponibili sono più di 1600 e vengono utilizzate per l'accesso ad ogni funzionalità della piattaforma. Oltre a poter essere ricercate mediante il loro transaction code queste transazioni sono anche elencate all'interno di

un menu, presente nella pagina principale di SAP, in modo da poter essere eseguite anche senza conoscerne l'identificativo.

Di seguito vengono illustrate alcune delle transazioni più utilizzate per lo sviluppo di programmi SAP ABAP, descrivendone le funzionalità ed il rispettivo transaction code. Queste transazioni sono raggruppate all'interno della voce di menu *ABAP Workbench*.

- **ABAP Dictionary (SE11)**: permette di leggere e scrivere dati all'interno delle tabelle del Database collegato al sistema SAP. E' possibile creare nuove tabelle, modificare la configurazione di quelle esistenti e cancellarle se necessario. Nel successivo capitolo di questo elaborato vengono, inoltre, discusse le modalità di interazione con il database all'interno di un programma ABAP.
- **ABAP Editor (SE38)**: uno degli applicativi maggiormente utilizzati durante lo sviluppo di programmi ABAP, rappresenta l'editor di testo integrato all'interno della piattaforma per la scrittura di codice. Permette di definire *function modules*, *screens* - ad esempio Dialog di un Dynpro - e tutti gli elementi che vanno a comporre un'applicazione SAP, utilizzando anche il meccanismo di **Forward Navigation**. Questo permette di analizzare la maggior parte degli elementi mostrati a schermo navigando verso la transazione appropriata mediante l'utilizzo del doppio click.
- **Function builder (SE37)**: permette di creare **Function modules**, ovvero moduli software che possono essere eseguiti da altri programmi mediante delle chiamate specifiche integrate nel codice. Questa applicazione offre anche la possibilità di definire l'interfaccia di input/output del modulo, definendone i parametri in ingresso e in uscita.
- **Menu painter (SE41) e Screen painter (SE51)**: vengono utilizzati per la costruzione delle interfacce grafiche di programmi Dynpro. Il primo gestisce dinamicamente le funzionalità e l'aspetto del menu di pulsanti visualizzato in alto all'apertura di un programma SAP. Il secondo, invece, permette la definizione degli elementi che costituiscono una schermata di un programma, sia in termini di aspetto che di funzionalità ed interazione con gli utenti.

- **Object Navigator (SE80):** rappresenta un punto di aggregazione degli strumenti utilizzati più di frequente nello sviluppo ABAP, tra cui quelli precedentemente elencati, creando un ambiente specifico, molto efficiente, per la programmazione. E' fondamentale nella gestione di grandi applicazioni composte da numerosi moduli, schermate ed oggetti di business logic.

1.2 SAP Extended Warehouse Management

La piattaforma SAP, come già detto in precedenza, è composta da numerosi prodotti capaci di comunicare tra loro mediante l'utilizzo di strumenti di trasferimento di documenti aziendali e informazioni di processo. Uno dei moduli più diffusi, soprattutto in ambienti come grandi impianti di stoccaggio o siti di produzione con annessi magazzini per la gestione dei processi di trasporto in entrata e in uscita, è SAP **Extended Warehouse Management**. Questo prodotto fa parte della più ampia suite di applicativi chiamata Supply Chain Management (SCM), indirizzata quindi alla gestione dei processi di trasporto, produzione e stoccaggio di beni all'interno di una realtà aziendale.

Come anche il nome suggerisce, SAP EWM viene utilizzato per la gestione delle operazioni di stoccaggio e movimentazione di prodotti all'interno di magazzini, più o meno automatizzati che siano. Rappresenta un'estensione del modulo *Warehouse Management*, integrato in ogni installazione SAP ERP, per espanderne le funzionalità e permetterne l'utilizzo in ambienti in cui i processi della catena logistica sono troppo complessi e articolati per la gestione semplificata offerta dal modulo base del software ERP.

Analizzando le capacità tecniche e le competenze dell'applicativo le principali caratteristiche sono:

- Impostazione della configurazione del magazzino secondo le migliori pratiche implementate dalla piattaforma. Questo comprende, ad esempio, la suddivisione dell'impianto in zone, corridoi, sezioni e unità di stoccaggio tenendo conto anche della locazione fisica di porte e zone di processamento in entrata e in uscita, garantendo che i percorsi dei beni all'interno dell'edificio siano ottimali per un risparmio sia in termini di costi che di tempo.

- Gestione dei processi in *inbound* e *outbound* - entrata e uscita dal sito di stoccaggio - dei beni immagazzinati. Questo comprende anche la capacità del software di dialogare con altri moduli, quali *Transportation management*, per la produzione di documenti di trasporto e bolle di carico, e altre componenti del pacchetto *SCM* per la gestione dell'intera catena di produzione.
- Gestione dei processi di ricezione e spedizione di beni, anche sul piano fiscale, in sinergia con il modulo SAP dedicato alla gestione finanziaria dell'azienda.
- Adempimento degli ordini presentati dai clienti avviando un iter, composto da processi sia contemporanei che consequenziali, per la distribuzione dei prodotti e l'evasione dell'ordine.
- Gestione automatizzata della creazione dei task operativi per le risorse umane dell'impianto, permettendo di suddividere in modo efficiente il carico di lavoro e ottimizzare i tempi.

Uno dei maggiori benefici derivanti dall'utilizzo di una soluzione così tecnologicamente avanzata è quello di poter gestire tutti questi processi in modo automatico una volta configurato ad hoc secondo le esigenze del singolo impianto di stoccaggio, grazie ad un sistema centralizzato di gestione capace di offrire anche numerosi strumenti di monitoraggio e controllo. Questi diventano fondamentali quando l'impianto raggiunge una certa dimensione ed anche la più semplice movimentazione di magazzino mette in moto una serie complessa di processi interdipendenti.

Infine si vogliono citare alcuni termini utili per la fruizione dell'elaborato:

- **Storage Bin:** unità minima di spazio all'interno del magazzino, può contenere beni stoccati in Pallet oppure sfusi.
- **Handling Unit (HU):** unità di gestione definita a sistema ed utilizzata per immagazzinare quantità multiple di un bene come un singolo elemento.

1.2.1 Comunicazione con il sistema ERP

Per garantire che venga sempre mantenuta la consistenza delle informazioni all'interno di tutti i moduli installati in una configurazione SAP ERP è

necessario che questi siano strettamente collegati e in grado di comunicare fra loro mediante un canale appositamente strutturato. In particolare, il trasferimento di dati dal sistema ERP a quello SCM può avvenire, a seconda del tipo di informazioni, mediante due protocolli differenti:

- **Core Interface (CIF):** uno dei metodi di comunicazione più utilizzato, permette di trasferire *Master Data* da un sistema ad un altro. Un esempio di *Master Data* è la definizione delle specifiche di un materiale all'interno del sistema ERP che deve essere trasportato in ambiente SCM per poter essere poi utilizzato da tutti i moduli appartenenti a questa soluzione.
- **Queued Remote Function Call (qRFC):** utilizzato, invece, per il trasferimento di documenti come ad esempio ordini di evasione o ricevute di ingresso merci. In questo caso il trasferimento avviene mediante l'utilizzo di documenti digitali contenenti le informazioni; questi documenti sono poi associati ad altri documenti di riferimento in ogni modulo e possono essere interpretati e tradotti.

1.3 Attività di inventario fisico

I processi di inventario fisico sono utilizzati, all'interno di un magazzino, per verificare lo stoccaggio corrente di prodotti sia ai fini di contabilità e controllo sia per mantenere il sistema informativo in linea con le effettive disponibilità. Questo procedimento è fondamentale per una corretta gestione di un magazzino in quanto l'aggiornamento delle informazioni riguardanti lo stato delle rimanenze ha un impatto sia sulla quantificazione dello stato patrimoniale dell'impresa sia sull'elaborazione dei risultati d'esercizio. L'unico scopo di questa attività inventariale è di accertare la presenza fisica dei prodotti all'interno del magazzino in modo da verificarne la corrispondenza con le giacenze contabilizzate e, in caso di discrepanze, permettere l'allineamento dei sistemi software di controllo mediante misure correttive.

In aggiunta, da un punto di vista normativo, è obbligatorio, in vista della chiusura di bilancio, rendicontare e valorizzare le rimanenze di magazzino per determinarne il valore fiscale nel caso in cui si verifichino due condizioni specifiche per due periodi d'imposta consecutivi [5]:

- Ricavi di cui all'art. 85 T.U.I.R., superiori ad Euro 5.164.568,99.

- Rimanenze finali di cui agli artt. 92 e 93 T.U.I.R., superiori ad Euro 1.032.913,80.

La schedulazione del processo di inventario fisico viene determinata scegliendo la modalità che risulta più adatta alle esigenze dell'impresa. In particolare, le pratiche maggiormente utilizzate nel mondo aziendale sono due:

- **Rendicontazione annuale:** in particolare nel periodo che va dal 24 Dicembre al 6 Gennaio, prevede la verifica dell'intero contenuto del magazzino in un'unica transazione. Questo processo viene solitamente eseguito quando le movimentazioni di magazzino sono ridotte al minimo, ad esempio nei periodi di chiusura aziendale.
- **Rendicontazione a rotazione:** prevede una conta parziale svolta a più riprese durante l'anno fiscale. La schedulazione dei conteggi può variare a seconda delle esigenze aziendali e ha l'obiettivo di verificare la presenza di ogni articolo stoccato nel magazzino almeno una volta all'anno.

1.3.1 Inventory Management and Physical Inventory

All'interno della suite di applicativi messi a disposizione da SAP ERP sono presenti anche gli strumenti necessari alla gestione dei processi di inventario di un magazzino. L'inventario fisico è un processo aziendale volto a mettere in relazione le quantità fisicamente presenti all'interno dell'impianto con i dati immagazzinati nella base di dati del sistema di gestione. Questo è un processo che deve essere eseguito obbligatoriamente per legge almeno una volta per anno fiscale; spesso, però, aziende di grandi dimensioni commissionano a terzi la gestione della contabilità dell'impresa e, solitamente, a seconda della complessità delle attività svolte vengono progettati ed implementati differenti meccanismi per lo svolgimento delle attività inventariali. Il processo di inventario fisico in SAP viene gestito mediante l'utilizzo di documenti digitali chiamati *Physical inventory documents* e può essere suddiviso in 3 fasi principali:

- Fase di preparazione: prevede la creazione del documento di inventario fisico, stampando e distribuendo agli operatori designati il documento

in modo che possano effettuare la conta ed eseguire il task a loro assegnato automaticamente dal sistema.

- Fase di conteggio: l'operatore si reca fisicamente presso gli spazi di magazzino individuati dal documento consegnatogli ed effettua una conta manuale dei beni presenti all'interno dell'unità analizzata.
- Fase di analisi: una volta terminato questo processo, i dati vengono salvati all'interno del documento ed il suo stato cambia, ad indicare che la conta è stata effettuata e il documento può essere finalizzato - operazione di *POST*. Durante questa fase può anche essere stabilita la necessità di effettuare nuovamente il conteggio, ad esempio a causa di una incongruenza delle informazioni con una differenza maggiore di una soglia preimpostata.

Una volta finalizzato il documento di inventario, questo non può più essere modificato e viene trasferito al modulo di gestione finanziaria per le dovute operazioni. Inoltre, viene creato un report delle discrepanze rilevate in fase di conteggio, in modo da permetterne l'analisi successivamente. In SAP esistono varie tipologie di documenti di inventario fisico utilizzate dal sistema. In questo elaborato verranno citate due di queste: documenti **HL**, che prevedono la conta ad hoc di tutti i prodotti contenuti in una *Storage Bin* o unità di stoccaggio, e documenti **HS**, che invece riguardano l'intero magazzino e si focalizzano su un singolo prodotto.

Capitolo 2

Tecnologie di riferimento

2.1 Il linguaggio ABAP

Il linguaggio utilizzato per la scrittura di applicativi all'interno dell'ambiente di esecuzione della piattaforma software SAP ERP, a partire dalla versione R/2, è ABAP, acronimo inizialmente utilizzato per abbreviare la dicitura Allgemeiner Berichts-Aufbereitungs Prozessor - Processore per la generazione di report generici - e negli anni successivi sostituito con Advanced Business Application Programming, contestualmente al rilascio di SAP R/3 in un tentativo di internazionalizzazione del prodotto.

ABAP, prendendo in considerazione la versione ABAP Objects estensione di ABAP/4, è un linguaggio di programmazione di quarta generazione (4GL) costruito specificatamente per la scrittura di applicazioni di business da eseguire all'interno dell'ambiente di runtime SAP. Rappresenta la base tecnologica su cui è costruita la piattaforma software: ad esclusione del componente fondamentale del sistema realizzata in C, tutti i SAP Business Modules, i componenti di sistema di SAP Basis e perfino l'ambiente di sviluppo interno sono scritti in ABAP.

In questo capitolo dell'elaborato si vogliono descrivere approfonditamente le caratteristiche fondamentali del linguaggio, le motivazioni alla base della sua nascita ed i principali strumenti messi a disposizione degli sviluppatori.

2.1.1 Caratteristiche del linguaggio ABAP

ABAP è un linguaggio che adotta un modello di programmazione ibrido. Questa sua caratteristica deriva dalla sua evoluzione nel tempo, essendo nato come linguaggio procedurale - con supporto a chiamate di procedure e gestione degli eventi di sistema - per poi essere esteso con i concetti della programmazione orientata agli oggetti. Entrambi i modelli sono interoperabili e possono coesistere all'interno di una applicazione. Il linguaggio è sia compilato che interpretato: il codice viene compilato da un compilatore integrato nella piattaforma in un formato intermedio per poi essere interpretato dall'ambiente di esecuzione ABAP in tempo reale.

ABAP è stato creato per essere adoperato esclusivamente nel processamento massivo di dati in applicazioni commerciali. Per questo motivo permette di utilizzare costrutti semantici - caratteristica propria dei linguaggi di quarta generazione e dei linguaggi di dominio specifico - in modo da codificare concetti complessi in modo elegante e conciso, al costo di una minore flessibilità al di fuori del proprio contesto operativo. A differenza di linguaggi senza un ambito applicativo, molte delle funzionalità avanzate del linguaggio sono direttamente incapsulate all'interno dello stesso e non fornite tramite librerie esterne mediante le quali estenderne le capacità; questo determina, da una parte, un aumento delle performance del linguaggio e la capacità di effettuare un controllo statico del codice migliore mentre, dall'altra parte, aumenta considerevolmente le parole chiave a disposizione degli sviluppatori e richiede un periodo di apprendimento più lungo per essere sfruttato al meglio.

Si evidenziano, di seguito, i principali aspetti presentati da ABAP e tipici dei linguaggi 4GL:

- Accesso al database integrato nell'ambiente di runtime ABAP mediante **Open SQL** e ottimizzazione delle performance grazie a **SAP Buffering**.
- Tabelle dinamiche *in memoria* per il processamento massivo di dati recuperati dal database.
- Accesso contemporaneo al database da parte di utenti multipli possibile grazie al componente **Online Transaction Processing**.

- Interfacciamento con altri ambienti di sviluppo con chiamate a funzioni remote (RFC).
- Interfacciamento integrato con XML.

Oltre agli strumenti integrati nel linguaggio vengono fornite numerose funzionalità grazie alle librerie presenti nel sistema come (1) gestione di oggetti in memoria condivisi e di oggetti persistenti all'interno del database, (2) accesso a Internet e (3) creazione delle interfacce utente. ABAP inoltre supporta la codifica testuale mediante Unicode.

Gestione dei dati

A causa della mole di dati da processare in tempo reale quando si utilizzano processi aziendali su larga scala uno dei punti cardine attorno ai quali è stato sviluppato ABAP è una gestione dei dati tale da ottimizzare le performance e velocizzare l'esecuzione dei programmi.

Uno degli strumenti adottati per raggiungere questo obiettivo è l'utilizzo delle **tabelle interne**, oggetti creati e mantenuti in memoria durante l'esecuzione del software all'interno dello spazio degli indirizzi ad esso riservato. La tabella interna è una struttura dati che il programmatore può, e deve, utilizzare, strutturandoli in modo adeguato per poter immagazzinare i dati di interesse recuperati dal database. In questo modo, le informazioni vengono richieste mediante una chiamata al server che ospita il database soltanto all'avvio del programma, permettendo di ridurre al minimo le operazioni di prelievo durante l'esecuzione in tempo reale del codice. Queste strutture vengono utilizzate anche per la memorizzazione temporanei di dati complessi calcolati a runtime. Questi possono, poi, essere inviati al database per un salvataggio permanente oppure eliminati poiché non più utili.

```
1 DATA :  
2   " TABELLA INTERNA  
3   l_tab_customer TYPE TABLE OF str_customer ,  
4   " WORK AREA  
5   l_str_customer TYPE str_customer .
```

Listing 2.1: Esempio di dichiarazione di una tabella interna

Le tabelle interne sono formate da una lista di *righe*, ognuna delle quali è a sua volta una struttura dati organizzata secondo le specifiche definite in fase di creazione della tabella - *colonne* - e contenente un valore per ognuno dei campi. L'accesso alle informazioni memorizzate all'interno delle tabelle interne avviene mediante una struttura di supporto, con le stesse specifiche della tabella, chiamata **Work Area**; in questo modo è possibile iterare sulla tabella andando a trasferire i dati di ogni riga nella work area ed elaborando poi le informazioni, senza agire mai direttamente sulla struttura dati principale. Anche nel caso di necessità di aggiungere una tupla alla tabella, questa verrà creata andando ad inserire i valori di interesse all'interno della work area e, successivamente, verrà creato un nuovo record con i contenuti di quest'ultima.

Nonostante questo strumento sia molto potente e permetta di gestire quantità massive di dati in maniera efficiente presenta, però, anche delle limitazioni: non è possibile creare tabelle multidimensionali e la dimensione è limitata a 2 GB. Di seguito si elencano i tipi di tabelle interne utilizzabili in un applicativo ABAP, mostrandone un esempio di dichiarazione per ognuna.

- **Standard table:** individuata dalla parola chiave *STANDARD* in fase di dichiarazione, viene gestita mediante un indice di riga logica. E' utilizzata per tabelle le cui tuple che dovranno essere elaborate sequenzialmente o con accesso alla singola riga mediante indice. Non essendo necessaria alcuna ulteriore computazione, queste tabelle sono le più veloci in fase di creazione. Supportano tutti i comandi di accesso ABAP: *INSERT*, *MODIFY*, *READ*, *DELETE*, *APPEND*, *COLLECT* e *LOOP*.

```

1 DATA :
2   l_tab_customer TYPE STANDARD TABLE OF
   str_customer .
```

Listing 2.2: Tabella standard

- **Sorted table:** tabella standard le cui righe, però, vengono ordinate mediante una o più colonne individuate con le parole chiave *NON-UNIQUE KEY* e *UNIQUE KEY*. Le nuove righe vengono inserite immediatamente nella posizione corretta in base all'ordinamento definito. Non supporta i metodi di accesso *COLLECT* e *APPEND*.

```

1 DATA :
2   l_tab_customer TYPE SORTED TABLE OF
   str_customer WITH NON-UNIQUE KEY key .

```

Listing 2.3: Tabella ordinata

- **Hashed table:** non possiede una indicizzazione delle righe, ma la ricerca viene eseguita mediante degli algoritmi di hashing che permettono di individuare la posizione della tupla di interesse con un tempo computazionale costante. Viene utilizzata in caso di accessi molto frequenti a singole tuple della tabella. Supporta gli stessi metodi di accesso delle tabelle ordinate.

```

1 DATA :
2   l_tab_customer TYPE STANDARD TABLE OF
   str_customer WITH NON-UNIQUE KEY key .

```

Listing 2.4: Tabella hashed

Di seguito si analizzano alcune delle parole chiave più utilizzate per l'accesso ai dati di una tabella interna.

- **INSERT:** inserisce una o più righe all'interno di una tabella. E' possibile specificare la posizione di inserimento in caso di tabella standard.

```

1 INSERT l_str_customer INTO l_tab_customer .
2 INSERT l_str_customer INTO l_tab_customer INDEX
   1 .

```

- **READ:** permette di leggere un record di una tabella. Se non è specificata una chiave per la ricerca, il primo elemento viene selezionato. I dati vengono copiati in una struttura di supporto dello stesso tipo.

```

1 READ TABLE l_str_customer WITH TABLE KEY name =
   'Georg Meyers' INTO l_str_customer .

```

- **LOOP:** permette di iterare sul contenuto di una tabella, andando a leggere un record alla volta. Copia i contenuti della tupla in una struttura di appoggio.

```

1 LOOP AT l_str_customer INTO l_str_customer.
2   l_str_customer-address-street_and_number = '
   Neurott Street 16'.
3 ENDLOOP.

```

- **MODIFY:** permette di modificare il contenuto di un record di una tabella, prendendo i dati da una struttura dello stesso tipo. E' possibile selezionare anche solo alcuni campi da modificare con la parola chiave *TRANSPORTING*.

```

1 MODIFY TABLE l_tab_customer FROM l_str_customer
   TRANSPORTING name INDEX 1.

```

Oltre alla gestione dei dati in memoria mediante costrutti come strutture e tabelle interne, a partire dalla versione Object di ABAP sono stati integrati anche meccanismi per l'accesso diretto ai dati memorizzati all'interno del database server. A differenza di altri linguaggi, dove per l'interazione con il database è necessario utilizzare un driver capace di stabilire una connessione e dialogare con il sistema di gestione delle tabelle, in ABAP è sufficiente utilizzare le parole chiave appositamente definite, data l'implementazione interna al linguaggio.

La pratica più diffusa per la tipizzazione delle variabili in applicativi ABAP non prevede l'utilizzo di *data types ABAP*, predefiniti dal linguaggio stesso, ma l'uso di *dictionary data types*. Questi rappresentano i tipi dei campi con cui sono definite le tabelle all'interno del database e permettono di effettuare un controllo più rigido in fase di dichiarazione di variabili, strutture e tabelle interne in un programma. In questo modo la corrispondenza fra i tipi dei campi di una struttura che deve, ad esempio, ospitare il contenuto di una tupla di una tabella e i tipi delle colonne della tabella stessa sarà sicuramente corretta. Inoltre, questo porta lo sviluppatore a poter utilizzare, in fase di scrittura del programma, tutti gli strumenti di aiuto messi a disposizione grazie all'analisi della definizione delle tabelle presenti nel database, semplificando notevolmente il lavoro.

Modularità

Leggibilità e modularità del codice sono due degli aspetti su cui si pone maggiore enfasi in fase di sviluppo di un programma, soprattutto se si prendono in considerazione software di grandi dimensioni, con decine di funzioni e centinaia o migliaia di diversi componenti che interagiscono tra loro. Anche ABAP mette a disposizione degli sviluppatori strumenti per semplificare la suddivisione del codice in componenti più piccole, con funzionalità e ambito di competenza limitati, ed in questo modo ridurre la ridondanza e rendere più leggibile e snello il codice prodotto.

Data la natura ibrida di ABAP, che presenta sia costrutti tipici dei linguaggi procedurali che elementi propri dei paradigmi orientati agli oggetti, sono diverse le possibilità messe a disposizione: appartengono alla prima categoria *FORM*, parola chiave del linguaggio per descrivere sotto-programmi con ambito operativo limitato, e *FUNCTION*, utilizzato per definire funzioni richiamabili mediante un'interfaccia esposta all'esterno; si guarda, invece, alla programmazione ad oggetti con *Classi* e *Interfacce*. Di seguito si analizzano in dettaglio gli elementi prima menzionati.

- **Sub-programs:** vengono definiti mediante le parole chiave *FORM* e *ENDFORM*, al cui interno viene racchiuso il codice che compone la routine. Utilizzando *USING* in fase di definizione del blocco di codice è possibile esplicitarne l'interfaccia, in modo che durante l'invocazione possano essere passati all'interno dell'ambiente i parametri necessari. Una routine può essere richiamata in qualsiasi momento all'interno di un programma mediante la parola chiave *PERFORM*, passando in ingresso eventuali parametri richiesti, come mostrato di seguito. Inoltre, un sottoprogramma può essere salvato all'interno di un file esterno ed incluso in un programma mediante la parola chiave *INCLUDE*.

```
1 PERFORM write_timestamp_using_screen USING
   l_timestamp.
2 FORM write_timestamp_using_screen
3   USING
4     i_timestamp TYPE timestampl.
5   WRITE i_timestamp TIME ZONE sy-zonlo.
6 ENDFORM.
```

Listing 2.5: Esempio di utilizzo di PERFORM

- **Function modules:** solitamente creati mediante interfaccia grafica utilizzando lo strumento *Function builder*, queste funzioni devono avere un identificativo univoco in tutto il sistema e presentare un'interfaccia parametrica ed un parametro di ritorno, oltre alla gestione di eventuali errori o eccezioni. Mediante lo strumento di creazione è possibile, altresì, corredare la funzione di una documentazione multilingua. I Function modules sono raggruppati, all'interno delle risorse di sistema, in *Function groups*, nei quali è possibile includere sia variabili locali utilizzabili da ogni componente sia altri oggetti, come schermate per l'interfaccia grafica. Di seguito si mostra un esempio di chiamata ad un Function module; in questo esempio viene mostrato l'uso della parola chiave *IMPORTING*, che definisce l'interfaccia dei parametri in ingresso, ma possono essere utilizzate anche le seguenti: *EXPORTING* per i parametri di ritorno, *CHANGING* per il passaggio per riferimento e *TABLES* per eventuali tabelle interne.

```
1 CALL FUNCTION 'getName'  
2   IMPORTING  
3     ev_name = l_name  
4   .
```

Listing 2.6: Function modules

- **Classes:** ABAP adotta, a partire dalla versione 4 del linguaggio, anche il paradigma di programmazione orientato agli oggetti mediante la propria implementazione di classi ed interfacce, integrando in aggiunta la gestione degli eventi di sistema. A causa della larga base di software già sviluppata negli anni, ABAP ha mantenuto la piena retro-compatibilità verso la precedente implementazione procedurale, creando un ambiente in cui entrambi i paradigmi possono essere usati nello stesso programma senza alcun problema. Come esempio in questo elaborato viene mostrata l'implementazione di una classe, sottolineando le scelte stilistiche adottate e la separazione netta fra dichiarazione ed implementazione.

```

1 CLASS static_vehicle DEFINITION.
2   PUBLIC SECTION.
3     CLASS-METHODS: accelerate IMPORTING delta TYPE
4       i,
5         show_speed.
6   PRIVATE SECTION.
7     CLASS-DATA speed TYPE i.
8 ENDCLASS.
9 CLASS static_vehicle IMPLEMENTATION.
10  METHOD accelerate.
11    speed = speed + delta.
12  ENDMETHOD.
13  METHOD show_speed.
14    DATA output TYPE string.
15    output = speed.
16    MESSAGE output TYPE 'I'.
17  ENDMETHOD.
18 ENDCLASS.

```

Listing 2.7: Esempio di definizione di una classe

2.1.2 Esecuzione di programmi ABAP in ambiente SAP

Ogni programma ABAP viene eseguito, all'interno di un sistema SAP, in un contenitore chiamato **Work Process**. Questi processi sono strutturati in modo da essere indipendenti dal sistema operativo in uso nello svolgimento delle proprie funzioni. Un Work Process è composto da 4 componenti:

- **Screen processor:** si occupa della gestione dell'interfaccia grafica creando, su richiesta, le schermate adatte e navigando all'interno delle finestre che compongono un programma.
- **Interprete ABAP:** componente che si occupa dell'esecuzione delle istruzioni di cui i programmi sono composti a partire da un codice intermedio creato, precedentemente, dal compilatore.
- **Interfaccia Database:** necessaria al dialogo tra il programma in esecuzione ed il server su cui viene istanziato il database.

- **Task handler:** gestisce la richiesta assegnatagli dal sistema andando ad operare sugli altri tre componenti secondo quanto codificato nel programma.

I processi vengono avviati e gestiti da un componente, il **SAP Dispatcher**, che si occupa della gestione di tutte le richieste che vengono sottoposte al sistema quali, ad esempio, l'avvio di un programma. E' compito del dispatcher assegnare la richiesta ad un Work Process in modo che questa possa essere soddisfatta.

Mentre il numero dei Work Process in un sistema viene configurato in fase di installazione non c'è limite al numero di utenti che possono connettersi ed interagire contemporaneamente; la distribuzione dei processi di sistema agli utenti dipende dalle richieste che questi ultimi producono e viene adattato in tempo reale cercando di mantenere efficiente e responsivo il sistema, garantendo all'utente un'esperienza soddisfacente e performante.

2.2 Framework per transazioni in radiofrequenza

Una transazione in radiofrequenza presenta poche, ma fondamentali, differenze rispetto ad una normale transazione SAP. In particolare, è una transazione che presenta una interfaccia utente altamente specializzata, concepita per essere mostrata su un dispositivo portatile che l'operatore di magazzino porta con sé ed utilizza per svolgere i compiti che gli vengono assegnati. Le operazioni, di movimentazione beni o conteggio di inventario ad esempio, vengono eseguite seguendo le indicazioni fornite dal terminale; eventuali informazioni significative vengono richieste e recuperate dal sistema centrale mediante lo stesso dispositivo che, solitamente, presenta uno schermo di dimensione prestabilita e un componente per la scansione di codici identificativi. La comunicazione tra il terminale ed il sistema informativo avviene mediante tecnologia in radiofrequenza appositamente configurata.

Il differente funzionamento di queste transazioni ha richiesto lo sviluppo di un ambiente di esecuzione apposito denominato **RF Environment**, accessibile mediante il codice transazione `/SCWM/RFUI`. All'interno di questa transazione SAP vengono forniti tutti gli strumenti necessari allo sviluppo e all'esecuzione di transazioni logiche RF. SAP mette a disposizione più di 80 transazioni RF integrate all'interno del sistema. Queste transazioni

coprono la maggior parte degli ambiti di utilizzo di una installazione SAP EWM e costituiscono la base per eventuali sviluppi di transazioni personalizzate secondo le esigenze del cliente. Anche la transazione sviluppata nel contesto di questo elaborato è stata progettata a partire dalla transazione RF standard per le attività di inventario fisico.

La struttura del framework RF impone una netta separazione tra logica applicativa e interfaccia utente. Infatti, la logica applicativa viene sviluppata come un normale programma ABAP a cui vengono, in seguito, collegate le schermate della transazione logica RF, progettate appositamente per il dispositivo di destinazione e la tipologia di utente che la andrà ad utilizzare. E' possibile, quindi, sviluppare numerose interfacce differenti per dispositivi che presentano schermi di diverse dimensioni e condividere fra loro l'intera logica di business.

Di seguito si elencano i termini e le definizioni fondamentali per avere una sufficiente comprensione delle transazioni logiche in radiofrequenza e l'ambiente di sviluppo *RF framework*.

- **Display Profile:** rappresenta un profilo che definisce la dimensione - in righe per colonne - dello schermo di un dispositivo ed il numero di pulsanti utilizzabili. Solitamente, in un magazzino sono presenti due tipi di schermi (e, di conseguenza, vengono utilizzati due Display Profile): monitor montati su carrelli elevatori, 8 righe per 40 colonne, e palmari con schermi più piccoli, 30 righe per 20 colonne. SAP mette a disposizione un Display Profile standarda, 8x40, individuato dalla stringa ******.
- **Application:** parametro che identifica il tipo dell'applicazione. L'applicazione è l'unità organizzativa del framework RF. In ambiente SAP EWM viene usato il codice **01**.
- **Presentation Profile:** utilizzato per specificare, in una transazione RF, logiche specifiche per un singolo magazzino. Il Presentation Profile standard ha il codice identificativo ********.
- **Personalization Profile:** definisce, per un gruppo di utenti, l'insieme delle transazioni RF da loro utilizzabili e come queste vengono organizzate nel menu di selezione. Ad ogni Presentation Profile è possibile associare uno o più Personalization Profile.

- **RF Environment:** ambiente di esecuzione delle transazione logiche RF. Una volta avviato, mediante il codice transazione /SCWM/RFUI, si viene accolti dalla schermata di logon in cui è necessario inserire il *codice di magazzino* SAP EWM, la *Risorsa* (identificativo del dispositivo) che si sta utilizzando ed il proprio **Presentation Device**.
- **Transazione logica RF:** è un programma ABAP, appositamente sviluppato per essere eseguito nell'ambiente *RF environment*. Solitamente viene avviato dal menu delle transazioni disponibile e consta di una o più schermate collegate tra loro. La logica della transazione viene suddivisa in **Step**, ad ognuno dei quali è associata una schermata e diversi *Function Code* e *Function Module*.
- **Application parameter:** strutture o tabelle ABAP che vengono mantenute dal framework RF e, al bisogno, passate ai Function Module di una transazione RF che ne fanno richiesta.
- **Function Code:** stringa lunga fino a 6 caratteri che associa un pulsante, fisico o virtuale che sia, ad un Function module della transazione capace di gestirlo. In questo modo, alla pressione di un pulsante il framework RF individua il Function Code collegato ed invoca la funzione ad esso associata.

Capitolo 3

Caso di studio

In questo capitolo si presentano nel dettaglio i requisiti dell'attività oggetto del presente elaborato, concordati fra l'azienda ed il cliente e formalizzati mediante numerose riunioni di pianificazione. Gli obiettivi posti sono stati determinati sulla base delle necessità aziendali del cliente e personalizzati secondo i regolamenti in vigore per le attività di inventario fisico. I dati raccolti vengono di seguito presentati in forma di Project Overview Statement in modo sufficientemente dettagliato da permettere di avere una panoramica generale dei bisogni espressi dal cliente e della soluzione individuata. In particolare, questa modalità di presentazione delle caratteristiche del caso di studio impone l'utilizzo della seguente struttura:

- Breve descrizione del problema che si vuole gestire o dell'opportunità di business che si vuole sfruttare.
- Presentazione dell'obiettivo finale del progetto.
- Suddivisione di quanto al punto precedente in sotto-obiettivi con l'obiettivo di evidenziare le macro-attività da affrontare.
- Assunzioni effettuate in fase di analisi per inquadrare al meglio il progetto e rischi individuati sulla base dei quali coordinare le attività di monitoraggio durante lo svolgimento del progetto.

Si lascia, invece, l'analisi dell'implementazione della soluzione all'interno della piattaforma SAP ai successivi capitoli.

3.1 Analisi del problema

L'affidamento della revisione dei conti dell'azienda cliente ad una nuova agenzia ha posto il problema di dover adeguare i processi utilizzati all'interno del magazzino per le attività di inventario fisico alle direttive imposte dalla nuova gestione in accordo con i regolamenti vigenti in materia. In particolare, è necessario personalizzare gli strumenti a disposizione degli operatori di magazzino all'interno della piattaforma software SAP in modo da permettere la semplificazione delle operazioni e l'implementazione di controlli più rigidi sui dati caricati all'interno del sistema.

3.2 Obiettivo del progetto

L'obiettivo è di codificare una transazione in radiofrequenza personalizzata utilizzabile dagli operatori di magazzino in completa autonomia per svolgere i processi di inventario fisico secondo le specifiche concordate ed inoltrare i documenti di inventario in modo automatico, senza la necessità di approvazione esterna in caso rispettino soglie prestabilite. Deve, inoltre, essere integrato all'interno del sistema un nuovo report destinato all'utilizzo per il monitoraggio delle attività inventariali, capace di mostrare i documenti che non hanno superato il controllo automatico dei conteggi e necessitano, quindi, di ulteriore supervisione.

3.3 Analisi dei requisiti della soluzione

- Implementazione di un Display Profile personalizzato per poter visualizzare correttamente le schermate della transazione RF sui dispositivi in dotazione agli operatori di magazzino.
- Restrizione delle operazioni permesse all'operatore addetto alle attività inventariali mediante la rimozione delle voci non utilizzate dal menu di scelta delle funzioni del dispositivo.
- Personalizzazione della transazione RF destinata all'inserimento dei conteggi per le operazioni di inventario fisico in modo da permettere una semplificazione delle operazioni.

- Recupero automatico di alcuni dei dati richiesti dal processo di conta dei prodotti a partire dalle informazioni memorizzate all'interno del database per evitare all'operatore di doverle inserire manualmente.
- Funzionalità di inoltro automatico dei documenti di inventario dal modulo EWM al sistema principale ERP sulla base del confronto dei conteggi con soglie predefinite.
- Realizzazione di un report personalizzato per la revisione dei documenti di inventario che sono stati bloccati in attesa di un controllo manuale in quanto contenenti misurazioni al di fuori dei limiti imposti in fase di configurazione. Da questo report dinamico deve, inoltre, essere possibile creare nuovi documenti di inventario da inviare agli operatori di magazzino per effettuare nuovi conteggi. In particolare, se il documento che non ha superato la verifica sulle differenze è di tipo *HL* e non è stata effettuata una riconta, deve essere generato un nuovo documento *HL*, con i rispettivi task di magazzino, utilizzando gli stessi dati di ricerca ed inserendo il riferimento all'attività fallita da cui è stato generato. Nel caso in cui anche la nuova operazione di conta non avesse successo, il documento generato sarà del tipo *HS* per il prodotto che ha generato l'errore, sempre che questo non sia già oggetto di un'altra operazione inventariale in atto; infine, nel caso anche questo documento non dovesse superare la fase di verifica, dovrà essere approvato manualmente dal revisore.

3.4 Criteri di successo

- Tempo di esecuzione di un task per il conteggio di un documento di inventario fisico ridotto del 60% a parità di condizioni.
- Riduzione degli errori degli operatori di magazzino in fase di inserimento dati del 90% grazie alla pre-compilazione dei campi e alla semplificazione dell'interfaccia grafica.

3.5 Assunzioni e rischi

- Durante l'analisi del problema sono state individuate alcune caratteristiche della soluzione sulla base dei comportamenti tenuti dagli operatori di magazzino durante le attività di conteggio.
- La soluzione è stata sviluppata tenendo conto del rischio di un possibile stravolgimento dei processi di inventario fisico nel futuro da parte dei revisori dei conti. Essendo questi determinati da una società esterna, che struttura le proprie operazioni tenendo conto anche delle possibili modifiche apportate ai regolamenti vigenti in maniera, il rischio di un precoce pensionamento della soluzione in favore di una nuova implementazione è stato tenuto in considerazione e valutato come trascurabile in relazione all'urgenza di implementazione del processo personalizzato.

Capitolo 4

Implementazione della transazione RF personalizzata

In questo capitolo verranno illustrate le operazioni effettuate per aggiungere la nuova transazione logica RF personalizzata all'interno del sistema SAP EWM già in uso da parte del cliente. E' possibile dividere, sommariamente, il processo in due fasi principali.

La prima prevede la personalizzazione delle impostazioni di sistema per poter codificare il dispositivo palmare utilizzato dagli operatori di magazzino; questo passaggio comprende anche la creazione, in copia da quella standard, dei componenti grafici della nuova transazione RF. La seconda fase, invece, illustra come è stata implementata la logica di funzionamento della transazione, comprendendo la creazione di un nuovo *Function Group* al cui interno sono stati raggruppati i *Function Module* dell'applicazione.

4.1 Operazioni preliminari

Prima di iniziare lo sviluppo sono state create due **Change Request** - in seguito CR - per il trasporto delle modifiche dal sistema di sviluppo **EWD** - sigla arbitraria per indicare il sistema Development del modulo EWM - verso quello di testing **EWQ**. Come già illustrato nel primo capitolo di questo elaborato, anche in questo caso la piattaforma SAP è stata configurata dal cliente scegliendo di declinarla in tre istanze separate, rispettivamente *Sviluppo*, *Controllo qualità e testing* e *Produzione*. Tutti gli sviluppi illustrati in questo e nel successivo capitolo della tesi sono stati effettuati nella

prima istanza.

Il primo passo è stato quindi quello di creare due CR per includervi tutte le modifiche da trasportare in seguito. Come è possibile vedere in figura 4.1, la creazione del pacchetto richiede l'inserimento di una descrizione del contenuto e la dichiarazione dell'utente del sistema responsabile della sua gestione. Questo processo risulta identico per qualsiasi tipologia di CR. In questo caso è stato utilizzato l'utente a me assegnato durante il progetto, *DEVELOPER*, e l'identificatore univoco assegnato automaticamente dal sistema all'atto della creazione è risultato essere **EMK900246** per la richiesta di tipo *Workbench*, contenente il codice dell'applicativo, mentre per quella *Customizing* è stato **EWMK900248**.

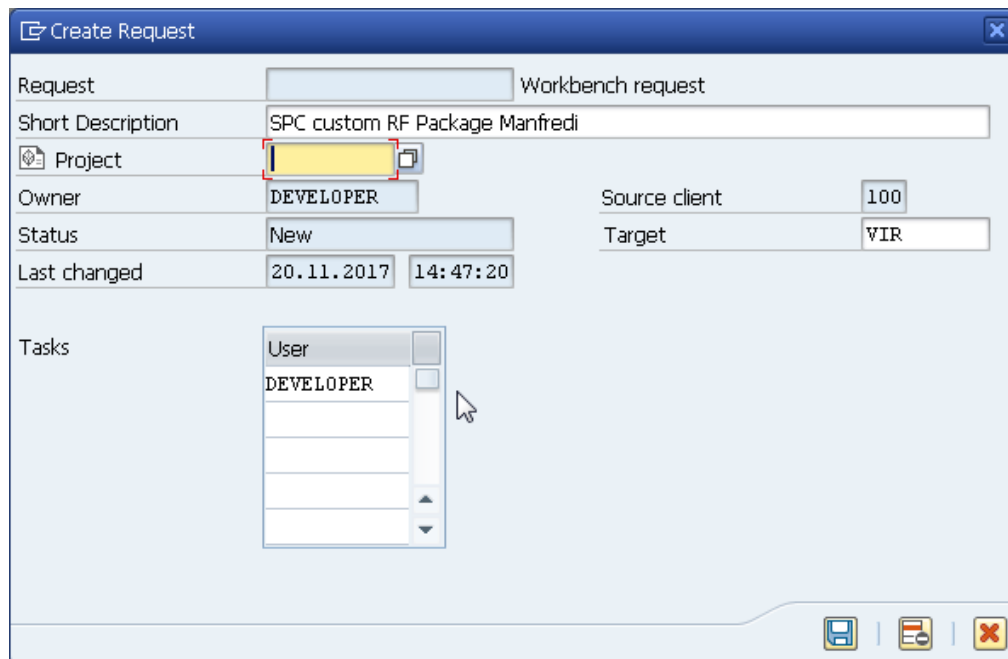


Figura 4.1: Creazione Change Request di tipo Workbench

Successivamente è stato configurato il **Package** di sviluppo del progetto. Questi contenitori sono utilizzati all'interno del sistema per collegare gli oggetti di sviluppo - *Function Modules*, *Function Groups*, *Screens* - con gli strumenti di organizzazione propri di *Application Server ABAP*. Tutti gli oggetti che possono essere modificati utilizzando l'ambiente di sviluppo ABAP *Workbench* devono obbligatoriamente far parte di un Package. E'

possibile navigare tra i Package definiti nel sistema mediante la transazione **Object Navigator (SE80)**, selezionando lo strumento *Repository Browser*. Qui si può, inoltre, creare un nuovo contenitore utilizzando l'interfaccia mostrata in figura 4.2.



Figura 4.2: Creazione Package di sviluppo

Seguendo la convenzione stabilita da SAP per le nomenclature di oggetti ABAP, il nome presenta come lettera iniziale "Z", a significare che è un oggetto personalizzato e non facente parte delle librerie SAP; "SPC" è, invece, un identificativo arbitrario del progetto di riferimento ed infine "MANF" indica brevemente l'utente responsabile, il sottoscritto. Le altre specifiche inserite all'interno del form di creazione sono standard.

4.2 Customizing

Come già illustrato nei capitoli precedenti, in un sistema SAP convivono due tipologie di personalizzazioni molto differenti. In primo luogo, tutti i programmi ABAP (Function module, librerie standard SAP, Screen, tabelle) che implementano logiche di processo strutturate ad hoc sulle necessità del cliente fanno parte delle impostazioni *Workbench*. Oltre a queste, esiste un'altra classe di impostazioni e variabili d'ambiente che regolano i processi funzionali che avvengono all'interno del sistema. Queste proprietà costituiscono il **Customizing** di un singolo sistema e possono essere modificate grazie alla transazione **SAP Project Reference Object (SPRO)**. Questa transazione presenta un menu ad albero mediante il quale selezionare le

proprietà di interesse. Questo menu ne facilita la ricerca suddividendole in cartelle, innestate su più livelli, in base al processo funzionale di riferimento. In riferimento al primo dei requisiti della soluzione da sviluppare, elencati nel capitolo 3 in fase di analisi del problema, la prima personalizzazione di *Customizing* è stata proprio quella di aggiungere un nuovo **Display Profile** nel sistema. Questo profilo è stato creato ad hoc secondo le specifiche dei dispositivi palmari in dotazione agli operatori di magazzino ed è stato associato, in seguito, ad un **Presentation Device**, che rappresenta il collegamento con la risorsa fisica all'interno della piattaforma SAP.

Il nuovo Display Profile, d'ora in avanti DP, è stato creato in copia da quel-

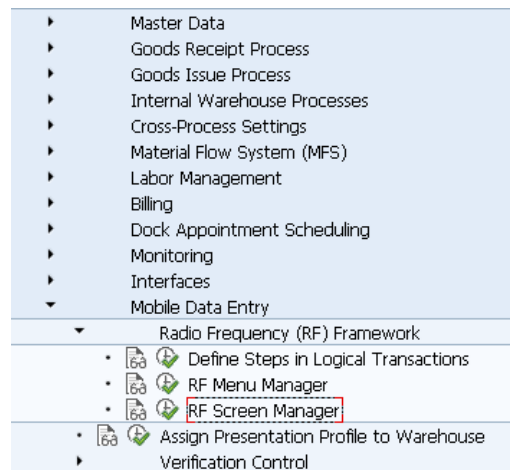


Figura 4.3: Ramo di Customizing per gestire l'interfaccia utente

lo standard SAP, codice identificativo **. La decisione di non modificare direttamente questo profilo, ma di crearne uno totalmente nuovo in copia da questo, è stata dettata dalle migliori pratiche in ambito di personalizzazione SAP: è universalmente consigliato di evitare di modificare, anche parzialmente, i componenti delle librerie standard installate a sistema, prediligendo sempre la creazione di nuove impostazioni ad hoc. Il nuovo DP presenta dimensioni dello schermo 16x20, a differenza di quello standard 8x40. In questa istanza, come ulteriore manovra preventiva, si è deciso di suddividere la creazione in due fasi: nella prima, si è creato un DP in copia dallo standard, senza però modificarne le dimensioni; a partire da questo, è stato creato il DP desiderato effettuando un'ulteriore copia.

Si mostra in figura 4.3 il ramo del menu di *Customizing* utilizzato per gestire

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF PERSONALIZZATA

il processo, oltre ad offrire una panoramica del menu di personalizzazione di Extended Warehouse Management. In particolare, il percorso della voce contenente tutte le impostazioni relative al framework RF è il seguente:

SAP Reference IMG -> SCM Extended Warehouse Management
-> Extended Warehouse Management -> Mobile Data Entry ->
Radio Frequency (RF) Framework

Una volta aperta la finestra **Screen Manager**, si viene accolti da un piccolo form che richiede di inserire il codice identificativo del DP che si vuole gestire. In figura 4.4 viene riportata la schermata di inserimento dati utilizzata per definire il nuovo DP e modificarne le dimensioni. Si è deciso di mostrare solo la seconda copia in quanto significativa ai fini dello sviluppo effettuato. Come è possibile notare, è stato creato il nuovo DP **Z5** - gli oggetti personalizzati dovrebbero sempre iniziare con la lettera Z - in copia da Z0, che come spiegato in precedenza è una replica di quello standard. Ci sono numerosi dettagli da evidenziare:

- All'interno della sezione *Screen Attributes* vengono definite le dimensioni effettive dello schermo su cui si dovranno visualizzare i dati.
- Nella sezione successiva vengono richiesti alcuni parametri propri dell'ambiente RF, ad esempio la lunghezza ed il numero di pulsanti da mostrare a schermo.
- Nella terza parte è necessario inserire i riferimenti ai *Function Group* che contengono la logica applicativa per la gestione del flusso delle schermate. In questo caso si è scelto il gruppo denominato **ZSPC_SCAR_RF_Z5_TMPL** che, non essendo ancora presente a sistema, verrà creato automaticamente come parte del processo.
- Infine, nella sezione relativa alle sotto-schermate del DP si è specificata la necessità di copiare anche queste e di convertirne automaticamente le dimensioni per rispettare il nuovo formato.

Un'ulteriore scelta di design, effettuata in fase di sviluppo, è stata la creazione di un *Function Group* diverso da quello legato al nuovo Display Profile, al cui interno verranno inseriti i *Function Module* utilizzati per l'elaborazione dei dati nella nuova transazione. Si è deciso di procedere in questo modo sostanzialmente per modificare interamente il flusso della tran-

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF PERSONALIZZATA

sazione standard ed evitare la creazione di conflitti con le variabili globali in fase di esecuzione. Il nuovo *Function Group* è stato denominato **ZSPC_RF_INVENTORY**.

The screenshot shows the 'Screen Manager' dialog box with the following configuration:

- SrcDispProfile: 20
- DstDispProfile: 25 (highlighted with a red box)
- Description: SCAR 16x20 Handheld
- Screen Attributes: Screen Height: 8, Screen Width: 20
- RF Screen Element Attributes: Pushb.Length: 8, Menu Item Lngth: 20, Pushb.Qty: 04, Message Display: 1
- Template Location: Function Group: ZSPC_SCAR_RF_Z5_TMPL, Templ.Scr.No.: 1, Function Group: ZSPC_SCAR_RF_Z5_TMPL, MsgTempl.ScrNo.: 0002, Function Group: (empty), Templ.Scr.Tit.: (empty)
- Sub-Screen Location: Create Sub-Screens, Function Group: ZSPC_SCAR_RF_Z5_TMPL, Convert Screens, Copy Screens

Figura 4.4: Creazione di un Display Profile in copia

Il passo successivo è stato quello di creazione del nuovo **Presentation Device** - d'ora in avanti PD - a cui associare il profilo appena creato. Co-

me già detto in precedenza, il PD rappresenta il nodo di collegamento tra la risorsa fisica utilizzata dagli operatori - inserite nel sistema come *Resource Device* - ed il DP adatto. In fase di *Logon* all'interno dell'ambiente RF l'operatore dovrà inserire il codice identificativo del proprio PD affinché il sistema possa recuperare il profilo associato alla risorsa e mostrare l'interfaccia utente adeguata. La gestione dei PD del sistema non fa parte delle transazioni comprese nelle attività di *Customizing* accessibili mediante *SPRO* ma è necessario usare la transazione */SCWM/PRDVC*. Questa altro non è che un'interfaccia per l'interazione con la tabella di sistema che mantiene i record di ogni PD. In combinazione con quest'ultima, è necessario utilizzare la transazione */SCWM/RSRC* per la creazione delle vere e proprie risorse utilizzate dagli operatori. Qui il campo *DefPresDvc* prevede la valorizzazione con il codice identificativo del PD.

In questo elaborato è stata definita ed utilizzata la risorsa **PI01**, appartenente al gruppo di risorse *ZCPI* dedicate alle operazioni di inventario fisico ed associate al PD da noi definito **ZCAR**.

L'ultimo passo necessario per concludere la creazione di una interfaccia personalizzata per una classe di dispositivi è la definizione delle transazioni accessibili da una determinata risorsa e come queste sono organizzate all'interno del menu principale. SAP utilizza due profili, il **Presentation Profile** e il **Personalization Profile**, per stabilire come il sistema deve comportarsi. In questo caso è necessario utilizzare il ramo di *Customizing Define steps in logical transactions*, sempre appartenente alla voce di menu relativa al framework RF, dove è possibile gestire tutte le proprietà, elencate nel capitolo 2, delle transazioni logiche RF presenti nel sistema, suddivise in un menu di navigazione laterale in modo da renderne più agevole la ricerca. La creazione dei due profili avviene in due fasi:

- Selezionando la voce *Define Presentation Profile* è possibile creare un nuovo profilo, in copia da quello standard ********, che nel nostro caso è stato rinominato **ZCAR**.
- Nella lista dei *Presentation Profile*, selezionando il profilo appena creato e cliccando sulla voce di menu **Define Personalization Profile**, è possibile gestire i profili di questo tipo associati. La schermata ripropone due informazioni, il campo *Application Parameter* e il *Presentation Profile* attualmente selezionati, e permette di definire un nuovo *Personalization Profile* - nel nostro caso chiamato **Z1**.

Mentre il *Presentation Profile* viene associato ad un magazzino intero, i *Personalization Profile* vengono collegati ai singoli utenti che utilizzano le risorse di sistema, permettendo di configurarne il menu di transazioni accessibili. In figura 4.5 vengono mostrati i profili appena creati. A questo

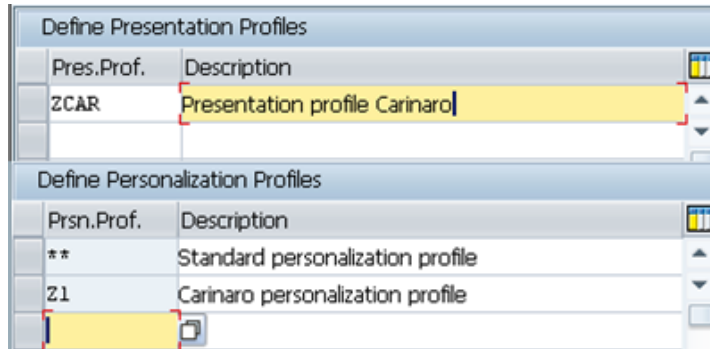


Figura 4.5: Presentation Profile e Personalization Profile

punto è necessario collegare il magazzino, nel nostro caso con codice identificativo **SCAR**, con il *Presentation Profile* appena definito, mentre occorre collegare ogni utente del sistema a tutte le risorse che caratterizzano la sue funzioni mediante la transazione `/SCWM/USER`. Nel nostro caso l'utente **DEVELOPER** è stato collegato al *Personalization Profile* **Z1**, gli è stato permesso l'accesso alle operazioni del magazzino **SCAR** utilizzando la risorsa **PI01**.

L'ultima fase di configurazione delle risorse è la definizione del menu di transazioni logiche in radiofrequenza mostrato all'utente dopo aver effettuato l'accesso all'ambiente di esecuzione *RF environment*, che ricordiamo essere codificato come `/SCWM/RFUI`. La possibilità di personalizzare il menu in base ai compiti di un operatore permette di stabilire con precisione quali sono le transazioni a cui l'utente può accedere, nascondendogli in questo modo le parti del sistema non di sua competenza. In questo caso, si è stabilito che le uniche transazioni accessibili devono essere quelle relative alle attività di inventario fisico e di movimentazione interna dei beni stoccati. Questa personalizzazione può essere configurata mediante la voce di *Customizing*, sempre inserita nel sotto-menu RF, chiamata **RF Menu Manager**. Anche in questo caso il menu è stato creato in copia a partire da quello standard offerto da SAP, associando il nuovo elemento ai profili creati precedentemente. Una volta terminata la creazione, sono state elimi-

nate le voci relative ai processi di entrata ed uscita merci dal magazzino, lasciando inalterate le altre opzioni; una volta terminata la creazione della nuova transazione logica RF questa verrà inserita nella relativa cartella.

4.3 Creazione della transazione ZIVCOU

La nuova transazione logica RF viene creata in copia da quella standard per le operazioni di conteggio del processo di inventario fisico. La transazione standard presenta il codice identificativo **IVCOUN**, derivato dalla sigla *Inventory Count*, e seguendo, come fatto finora, le buone pratiche nella scelta delle nomenclature ABAP è stato deciso di assegnare il nome **ZIVCOU** alla nuova transazione. La transazione ha sempre come obiettivo quello di effettuare la conta di un documento di inventario fisico assegnato all'operatore mediante un task apposito ma, sulla base dei requisiti comunicati dal cliente, verranno apportate delle modifiche alle schermate dell'applicativo variandone sia l'aspetto che le funzioni.

Mediante la creazione in copia da quella standard viene automatizzato il processo di assegnazione dello *Step* di partenza, quello richiamato all'avvio della transazione, ed il framework RF propone di copiare anche tutte le sue dipendenze come mostrato in figura 4.6. In questo caso specifico il numero di dipendenze copiate è stato di 1336 e la transazione *ZIVCOU* rappresenta, in questa fase, una copia esatta di quella standard. L'ultimo passaggio da

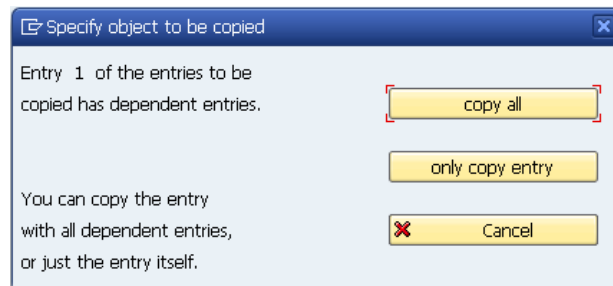
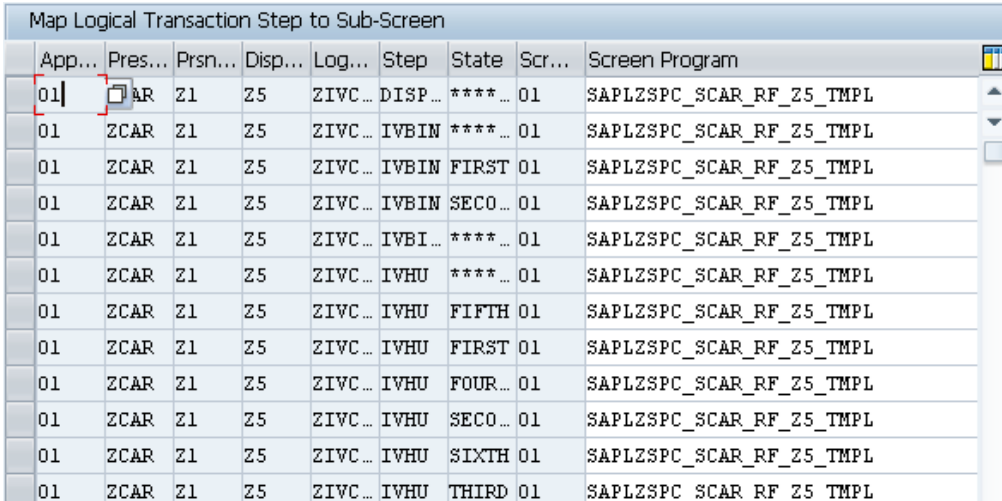


Figura 4.6: Dialog di copia delle dipendenze della transazione di partenza

effettuare prima di poter cominciare a lavorare alla modifica delle schermate della transazione custom è l'associazione delle sotto-schermate appena create con il *Presentation Profile* ZCAR da noi definito. All'interno della

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF PERSONALIZZATA

finestra *Map Logical Transaction Step to Sub-screen* viene presentata allo sviluppatore una tabella con un tupla per ogni sotto-schermata; questa tabella permette la definizione del processo funzionale che le caratterizza.



App...	Pres...	Prsn...	Disp...	Log...	Step	State	Scr...	Screen Program
01	ZCAR	Z1	Z5	ZIVC...	DISP...	****	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVBIN	****	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVBIN	FIRST	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVBIN	SECO...	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVBI...	****	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVHU	****	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVHU	FIFTH	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVHU	FIRST	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVHU	FOUR...	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVHU	SECO...	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVHU	SIXTH	01	SAPLZSPC_SCAR_RF_Z5_TMPL
01	ZCAR	Z1	Z5	ZIVC...	IVHU	THIRD	01	SAPLZSPC_SCAR_RF_Z5_TMPL

Figura 4.7: Finestra di mappatura delle sotto-schermate

In ordine, in riferimento alla figura 4.7, i campi della tabella sono quelli elencati di seguito:

- Application
- Presentation Profile
- Personalization Profile
- Display Profile
- Logical transaction
- Step
- State - utilizzato come punto di controllo intermedio all'interno di uno Step.
- Screen

- Screen Program - Function group al cui interno sono raggruppate le schermate del Display Profile selezionato.

Inizialmente, queste tuple create con il meccanismo di copia della transazione logica fanno riferimento al *Presentation Profile* standard ****. E' necessario quindi copiare manualmente tutte le tuple della transazione logica di interesse, in questo caso *ZIVCOU*, ed associare le copie al *Presentation Profile* personalizzato *ZCAR*, come si può vedere nella figura.

4.3.1 Creazione delle schermate personalizzate

I requisiti discussi nel capitolo 3 prevedono che le schermate associate alle operazioni di conta vengano modificate per compilare automaticamente alcuni campi e nascondere del tutto altri. Gli *Step* della transazione logica adibiti a questa operazione sono i seguenti:

- **IVMHU**: conteggio dei prodotti contenuti all'interno delle *HU* - solitamente Pallet - presenti all'interno di una *Storage bin*.
- **IVMOH**: conteggio dei prodotti sfusi presenti all'interno di una *Storage Bin*.
- **IVMHUV**: conteggio di tutte le *HU*, registrate nel sistema informativo, contenenti un determinato prodotto.
- **IVMOHV**: conteggio di tutte le *Storage Bin* contenent un determinato prodotto.

Ognuno di questi *Step* è associato ad una schermata, rispettivamente 53, 55, 54 e 56. Tutte queste schermate sono state modificate secondo gli stessi requisiti; per questo motivo si riporta qui esclusivamente il procedimento eseguito per la modifica della schermata 53. Secondo la convenzione delle nomenclature in ABAP, tutte le schermate personalizzate di un programma devono essere numerate in modo decrescente, a partire dal numero 9999. Per questo motivo si è scelto di numerare la nuova schermata 9999 e le altre, rispettivamente, 9998, 9997 e 9996.

La creazione di un nuovo *Screen* ABAP è possibile utilizzando la transazione **SE80**, corrispondente a *Object Navigator*. Qui è stato aperto il *Function*

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF PERSONALIZZATA

Group collegato al nostro *Display Profile* e, nella sezione **Screens** del menu di navigazione, mediante il tasto destro è stata copiata la schermata 53 nella schermata 9999 dello stesso pacchetto. A questo, con un doppio click, è possibile aprire l'oggetto appena creato e, successivamente, avviare il programma **Screen Painter** mediante il tasto *Layout*. Lo strumento di modifica si presenta come in figura 4.8. Com'è possibile notare, la schermata

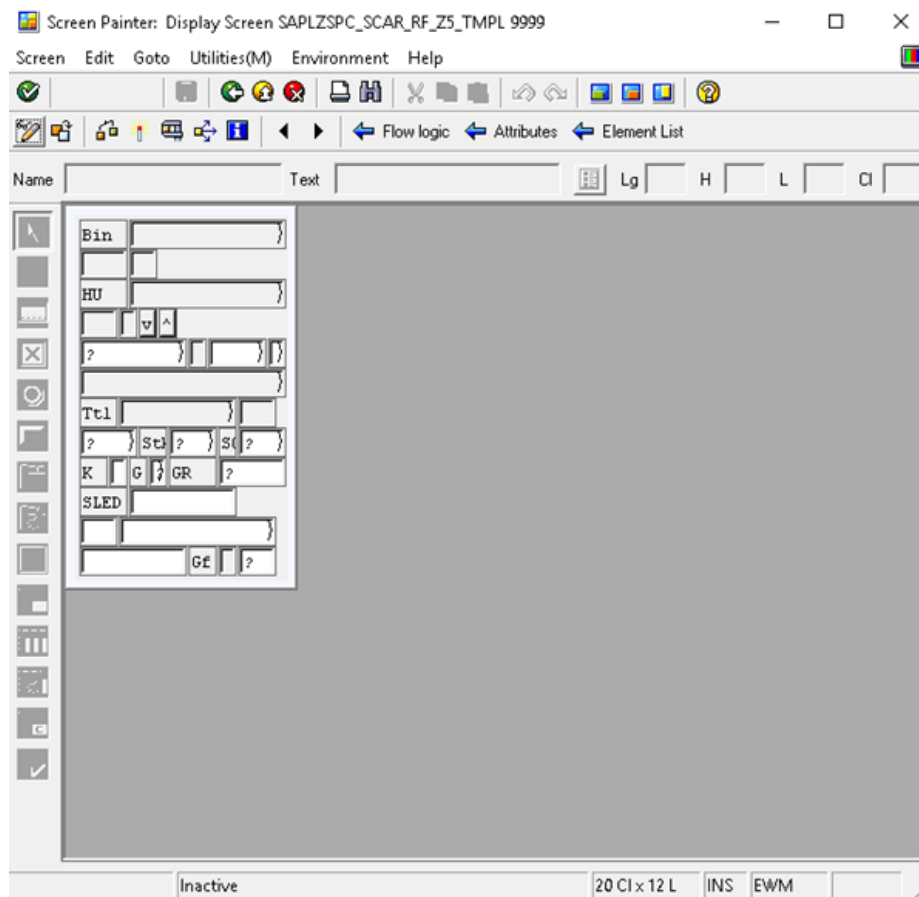


Figura 4.8: Schermata 9999 analizzata con lo strumento Screen Painter

ta presenta numerosi campi che vengono visualizzati in modo non corretto, conseguenza diretta della conversione automatica delle sotto-schermate in fase di copia del *Display Profile*.

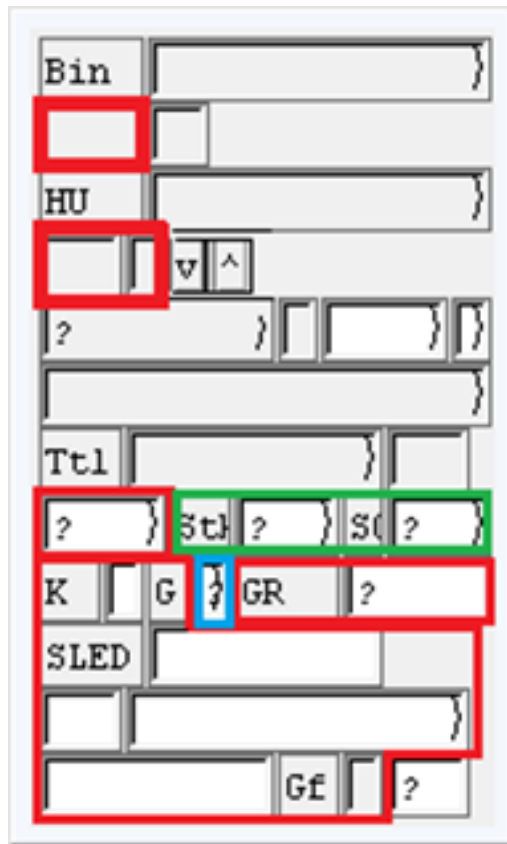


Figura 4.9: Diagramma delle modifiche apportate alla schermata

Per ovviare a questo problema e soddisfare le richieste del cliente, sono state previste le seguenti modifiche, mostrate anche graficamente in figura 4.9:

- I campi in rosso sono stati eliminati in quanto non utilizzati dagli operatori di magazzino.
- I campi evidenziati in verde, destinati all'inserimento manuale del magazzino proprietario del bene e dell'entità autorizzata a disporne, vengono nascosti all'utente e calcolati in modo automatico.
- Il campo evidenziato in blu, corrispondente alla categoria di stoccaggio del bene scansionato, rimane visibile e non viene più valorizzato dall'operatore ma dal sistema autonomamente.

- Gli altri elementi della schermata sono stati ridimensionati ed organizzati diversamente nella pagina in modo da rendere il contenuto della schermata più facile da leggere.

Il risultato finale è quello mostrato in figura 4.10 e rispecchia, allo stesso modo, quanto fatto con le schermate 9998, 9997 e 9996.

The screenshot shows a vertical form with the following elements from top to bottom:

- A label **Bin** followed by a text input field and a small square button.
- A horizontal separator line.
- A label **HU** followed by a text input field and two small square buttons with downward and upward arrow symbols.
- A horizontal separator line.
- A label **Product** followed by a text input field containing a question mark, a closing curly brace, and a small square button.
- A label **Quantity** followed by a text input field, a closing curly brace, and a small square button.
- A horizontal separator line.
- A label **Total** followed by a text input field, a closing curly brace, and a small square button.
- At the bottom, four small square buttons arranged in a row.

Figura 4.10: Schermata 9999 definitiva

A questo punto sono state modificate le mappature delle schermate agli **Step** della transazione logica *ZIVCOU*, sostituendo, all'interno della finestra di configurazione *Map Logical Transaction Step to Sub-screen*, i riferimenti alle precedenti con quelli appena creati.

4.4 Implementazione delle modifiche funzionali

In questa sezione verrà analizzata in dettaglio l'implementazione delle modifiche funzionali richieste dal cliente per personalizzare le operazioni di conteggio per l'inventario fisico. In particolare, è possibile dividere l'operazione in due fasi separate:

- Nella prima fase, sono state modificati 4 *Function module*, relativi al completamento automatico di alcuni campi della schermata sulla base del codice prodotto scansionato. Il processo standard prevede che sia l'operatore ad inserire queste informazioni manualmente mentre, nella versione modificata, verranno recuperate in modo automatizzato in corrispondenza della pressione del tasto **ENTER** successivamente all'inserimento, o alla scansione, del codice identificato del bene analizzato.
- La seconda fase, invece, mostra la logica computazionale aggiunta al processo di conteggio per far sì che i documenti di inventario, al momento del salvataggio, non vengano semplicemente segnati con lo stato **COUNT**; le informazioni registrate al loro interno verranno, infatti, confrontate in tempo reale con dei dati di riferimento per stabilire se il documento può essere approvato senza ulteriore verifica. In caso le differenze superino determinate soglie preimpostate, il documento dovrà essere cancellato - stato **DELE** - e le informazioni in esso contenute saranno salvate in una nuova tabella custom per poi, in seguito, venire revisionate manualmente.

I *Function Module* collegati ad un'applicazione con interfaccia grafica sono di due tipologie: **PBO** - Process Before Output - e **PAI** - Process After Input. I primi vengono invocati prima che la schermata sia popolata con i dati necessari ed hanno l'obiettivo di permettere che le informazioni vengano recuperate dal database ed elaborate prima di essere mostrate all'utente. I *Function Module* del secondo tipo, invece, sono invocati ogni volta che un input viene recepito dall'ambiente di esecuzione; ad esempio, è possibile inserire all'interno di un modulo PAI un segmento di codice da eseguire ogni volta che l'utente preme il pulsante corrispondente, ad esempio nelle transazioni logiche RF, al *Function Code* "ENTER". Come già illustrato

nel capitolo 2, un *Function Code* rappresenta l'associazione tra un pulsante, fisico o virtuale, ed il rispettivo modulo funzionale contenente la logica applicativa per la gestione della sua pressione in base al contesto di esecuzione.

E' inoltre necessario distinguere, in caso di un'applicazione che utilizzi i *Dialog* come ad esempio un programma *Dynpro* o una transazione RF, tra moduli *PBO/PAI* collegati ad un *Function Module* o a una schermata. L'interazione fra queste due categorie viene mostrata in figura 4.11.

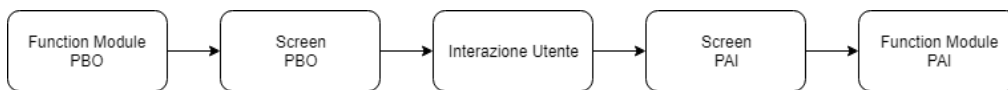


Figura 4.11: Flusso PBO-PAI in transazione RF

Inoltre, si vuole far notare il procedimento necessario per l'associazione di un *Function Module* personalizzato ad uno *Step* di una transazione logica in radiofrequenza. All'interno della rispettiva schermata di *Customizing* è presente la sezione denominata **Define Logical Transaction Step Flow**. Selezionandola viene mostrata una tabella contenente i seguenti campi:

- *Presentation Profile*
- *Logical Transaction*
- *Step*
- *Function Code*
- *Function Module*: modulo funzionale invocato all'attivazione del *Function Code*.
- *Next Step*: prossimo *Step* da eseguire al termine dell'esecuzione del *Function Module*.
- *Processing Mode*: modalità di processamento; i valori accettati sono: **Foreground**, con visualizzazione a schermo del risultato, **Background**, in caso l'elaborazione non produca alcun output, e **Defined during execution** che lascia al modulo il compito di definire come deve avvenire l'esecuzione.

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF PERSONALIZZATA

- *Background Function Code*: Nel caso in cui il processamento avvenga in secondo piano è necessario specificare anche il successivo *Function Code* da attivare al termine dell'elaborazione.

I primi quattro campi compongono la chiave primaria della tabella, individuando univocamente il singolo *Function Code* di uno *Step* della transazione. Ogni tupla della tabella rappresenta una transizione da uno *Step* ad un altro mediante l'esecuzione di un *Function Module*, invocato all'attivazione di un *Function Code*. Per ogni transizione vengono quindi dichiarati lo *Step* successivo ed il tipo di processamento desiderato. Se ne mostra in figura 4.12 un esempio.

Define Logical Transaction step flow							
Pres.P...	Log.Tr...	Step	Func.C...	Function Module	Next Step	Proc.Mode	BckgrFCode
ZCAR	ZIVCOU	IVMHU	ENTER	/SCWM/RF_MAT_IVMHU_PAI	IVMHU	0 Defined during execu...	
ZCAR	ZIVCOU	IVMHU	MAT	/SCWM/RF_MAT_IVMHU_PBO	IVMHU	2 Foreground	

Figura 4.12: Definizione del flusso di una transazione RF

4.4.1 Compilazione automatica dei campi

Il procedimento descritto di seguito è stato ripetuto in egual modo per i 4 *Function Module* modificati. In particolare, le modifiche sono state inserite nel *Function Module PAI* dei 4 *Step* elencati precedentemente: *IVMHU*, *IVMOH*, *IVMHUV* e *IVMOHV*. Si mostra, come esempio, il modulo `ZSPC_RF_MAT_IVMHU_PAI_M` relativo al caso in cui si stia processando un documento di inventario *HL* con presenza di HU all'interno dell'ubicazione.

```
1 " Importazione delle variabili globali d'esecuzione
2 CALL FUNCTION '/SCWM/RF_INV_GET_GLOBVAR'
3   IMPORTING
4     ev_lgnum          = gv_lgnum
5     ev_who            = gv_who
6 *   EV_TIMEZONE      =
7     ev_guided_inventory = gv_guided_inventory
8 *   EV_INVENTORY_NOT_POST_ITEM =
9 *   EV_IND_DOC_TYPE  =
```

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF
PERSONALIZZATA

```
10 *     EV_IND_DELIVERY      =
11 *     EV_WITH_HU          =
12     ev_level              = gv_level
13 *     EV_CURRENT_LINE_HEAD =
14 *     ES_STEP              =
15 *     ES_COUNT_FOR_CW     =
16 *     ES_STOCK_FOR_CW    =
17 *     ES_QUAN_FOR_CW     =
18 *     ES_SN_FOR_CW       =
19 *     ET_HU_STAT          =
20 *     ET_BUNDLE           =
21 *     ET_QUAN             =
22 *     ET_SN               =
23     eo_cust               = grefo_cust
24 *     EO_CUST_SCWM        =
25     eo_rfpi               = grefo_rfpi
26     eo_pack               = grefo_pack
27     eo_ui_fields         = grefo_ui_fields
28     eo_stock_id          = go_stock_id
29 *     EO_PI_SELECTS       =
```

Listing 4.1: Importazione delle variabili d'esecuzione globali

Viene qui mostrato un estratto di codice contenente l'invocazione di un *Function Module* che ha l'obiettivo di importare nel modulo corrente le variabili globali che caratterizzano il contesto di esecuzione. Per brevità, sono state omesse le dichiarazioni di variabili, con la rispettiva tipizzazione. Di seguito, invece, viene integrato il codice responsabile della compilazione automatica dei campi.

```
1 " Creazione oggetto per il monitor stock - contiene
   i dati di stoccaggio correnti
2 IF lo_mon_stock IS NOT BOUND.
3 CREATE OBJECT lo_mon_stock EXPORTING iv_lgnum =
   gv_lgnum.
4 ENDIF.
5
6 " Recupero la bin corrente dalla sezione di testa
   del documento di inventario
```

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF
PERSONALIZZATA

```
7 CLEAR: ls_head, lv_lgpla.
8 READ TABLE ivheadtab INTO ls_head INDEX 1.
9 MOVE ls_head-lgpla_sel TO lv_lgpla.
10
11 " Creo le strutture necessarie alla ricerca
12 REFRESH lt_matnr_r.
13 CLEAR ls_matnr_r.
14 ls_matnr_r-sign = wmegc_sign_inclusive.
15 ls_matnr_r-option = wmegc_option_eq.
16 ls_matnr_r-low = ls_count-matnr_verif.
17 ls_matnr_r-high = ls_count-matnr_verif.
18 APPEND ls_matnr_r TO lt_matnr_r.
19
20 REFRESH lt_lgpla_r.
21 CLEAR ls_lgpla_r.
22 ls_lgpla_r-sign = wmegc_sign_inclusive.
23 ls_lgpla_r-option = wmegc_option_eq.
24 ls_lgpla_r-low = lv_lgpla.
25 ls_lgpla_r-high = lv_lgpla.
26 APPEND ls_lgpla_r TO lt_lgpla_r.
27
28 "Recupero i dati dello stock dal monitor
29 CALL METHOD lo_mon_stock->get_stock_overview
30 EXPORTING
31     it_matnr_r      = lt_matnr_r
32     it_cat_r        = lt_cat
33     it_owner_r      = lt_owner
34     it_entitled_r   = lt_entitled
35     it_charg_r      = lt_charg
36     it_lgtyp_r      = lt_lgtyp
37     it_lgber_r      = lt_lgber
38     it_lgpla_r      = lt_lgpla_r
39     it_lptyp_r      = lt_lptyp
40     it_aisle_r      = lt_aisle
41     it_stack_r      = lt_stack
42     it_level_r      = lt_level
43     it_binsc_r      = lt_binsc
```

```

44     it_depth_r           = lt_depth
45     it_psa_r            = lt_psa
46     it_rsrc_r          = lt_rsrc
47     it_tu_num_ext_r     = lt_tuext
48     it_tsp_r           = lt_tsp
49     IMPORTING
50     et_stock_overview = lt_stock_overview
51     et_physical_stock = lt_phy_stock
52     ev_error          = lv_error.
53
54     " Inserisco i dati recuperati all'interno dei campi
55     della schermata
56     READ TABLE lt_stock_overview INTO ls_stock_overview
57     INDEX 1.
58     IF ls_stock_overview IS NOT INITIAL.
59         IF ls_stock_overview-cat IS NOT INITIAL.
60             ls_count-cat = ls_stock_overview-cat.
61             ls_count-owner = ls_stock_overview-owner.
62             ls_count-entitled = ls_stock_overview-
63             entitled.
64         ENDIF.
65     ENDIF.
66
67     IF ls_count IS NOT INITIAL.
68         MODIFY ivmattab FROM ls_count INDEX lv_line.
69     ENDIF.

```

Listing 4.2: Compilazione automatica dei campi della finestra

La compilazione dei campi mostrati a schermo è possibile grazie ad un'interfaccia, integrata nel framework, che automaticamente lega i campi di input della finestra ad una struttura dati accessibile all'interno del *Function Module*. All'attivazione del *Function Code* "ENTER" il modulo controlla che sia stato scansionato un materiale e, in caso positivo, tramite un oggetto ABAP che permette l'accesso al **Monitor stock** è possibile recuperare i dati di sistema del materiale per compilare in modo automatico alcuni campi dell'interfaccia grafica. Una volta che le informazioni - *Stock type*, *Owner* e *Entitled* - sono stati recuperate vengono trasferite all'interno della strut-

tura dati *ivmattab* affinché siano disponibili anche per i successivi moduli funzionali.

Il codice mostrato sopra permette di avere una dimostrazione di come tutti i dati inseriti all'interno del sistema informativo possono essere recuperati da qualsiasi *Function Module* mediante gli oggetti e le strutture dati preposte. Nella prossima sezione viene illustrato, invece, il lavoro svolto per l'implementazione della funzionalità di posting automatico dei documenti d'inventario contati.

4.4.2 Implementazione della funzionalità di conferma automatica dei documenti di inventario

Questa sezione dell'elaborato di tesi ha l'obiettivo di mostrare uno dei componenti fondamentali dello sviluppo di un programma ABAP: la possibilità di definire delle tabelle personalizzate all'interno del database per salvare eventuali parametri di funzionamento di programmi custom o informazioni sui processi funzionali dell'azienda. Per sviluppare la funzionalità di posting automatico dei documenti richiesta dal cliente, infatti, sono state integrate nel sistema due tabelle personalizzate:

- **ZSPC_PARM:** tabella utilizzata da tutti i programmi ABAP custom realizzati per il progetto *SPC*, permette di configurare l'ambiente di esecuzione senza dover modificare il codice degli applicativi.
- **ZSPC_PHY_DOC:** tabella condivisa dal *Function Module* responsabile per il posting automatico ed il Report di gestione dei documenti cancellati. All'interno di questa tabella vengono inseriti i dati dei documenti cancellati a causa di una differenza tra i valori conteggiati e le rimanenze di magazzino maggiore di una soglia preimpostata. Queste informazioni vengono poi utilizzate dal programma Report per, eventualmente, creare nuovi documenti di inventario secondo dei criteri prestabiliti.

Di seguito viene analizzato in dettaglio la struttura delle due tabelle.

Tabella ZSPC_PARM

La chiave per individuare una tupla all'interno della tabella è composta dai seguenti campi:

*CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF
PERSONALIZZATA*

Field	Key	Ini...	Data element	Data Type	Length	Deci...	Short Description
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3		0 Client
ZPROC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ZPROC	CHAR	20		0 Procedure
REPID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	REPID	CHAR	40		0 ABAP Program Name
ZPARM	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ZPARM	CHAR	20		0 Parameter ID
ZPROG	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ZPROG	NUMC	2		0 Progressive Number
ZSIGN	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZSIGN	CHAR	1		0 Include/Exclude
ZOPTN	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZOPTN	CHAR	2		0 Option
ZLOW	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZLOW	CHAR	40		0 Minimum value
ZHIGH	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZHIGH	CHAR	40		0 Maximum value
ZNOTA	<input type="checkbox"/>	<input checked="" type="checkbox"/>		CHAR	80		0 Note
ZUTIL	<input type="checkbox"/>	<input checked="" type="checkbox"/>		CHAR	1		0 X=Used parameter
ZLPGM	<input type="checkbox"/>	<input checked="" type="checkbox"/>		CHAR	40		0 Last program which used the parameter
DATUM	<input type="checkbox"/>	<input checked="" type="checkbox"/>		DATS	8		0 Last used date
UZEIT	<input type="checkbox"/>	<input checked="" type="checkbox"/>		TIMS	6		0 Last used time
ZTPAR	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZTPAR	CHAR	1		0 Parameter type
ZTAGG	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZTAGG	CHAR	2		0 Update type
ZFAGG	<input type="checkbox"/>	<input checked="" type="checkbox"/>		INT1	3		0 Update Frequency
ZTFLD	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZTFLD	CHAR	1		0 Data type
ZDFLD	<input type="checkbox"/>	<input checked="" type="checkbox"/>		INT1	3		0 Number of decimals places

Figura 4.13: Tabella custom ZSPC_PARM

- **MANDT**: codice mandante del client utilizzato, determina i dati di interesse in base al profilo dell'account che si adopera per il login al sistema.
- **ZPROC**: campo utilizzato per stabilire l'ambito operativo della procedura che si sta utilizzando.
- **REPID**: nome del programma ABAP che utilizza il parametro.
- **ZPARM**: identificativo del parametro.
- **ZPROG**: campo numerico progressivo utilizzato per salvare un range di valori per uno stesso parametro.

Gli altri campi della tabella permettono di parametrizzare differenti tipi di strutture dati. Ad esempio i campi **ZSIGN-ZOPTN-ZLOW-ZHIGH** permettono, se usati in combinazione, di selezionare un range di valori in caso si stia usando una struttura ABAP del tipo **RSELOPTION**. Inoltre, un altro campo da evidenziare è **ZUTIL**, booleano in grado di attivare o disattivare una tupla della tabella in fase di configurazione.

All'interno di questa tabella, utilizzata anche da altri programmi ABAP custom del sistema, sono state inserite le tuple mostrate in figura 4.14. Per

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF PERSONALIZZATA

ogni tupla la tripletta mancante per la composizione della chiave è composta dai valori **100 - SCAR - ZSPC_INVE_AUTO_POST**.

Ogni tupla regola il comportamento della funzione di posting automatico relativamente ad una tipologia di sezioni di stoccaggio del magazzino. Il campo *ZLOW* indica la soglia, in percentuale, con cui confrontare la differenza rilevata in fase di conteggio.

ZPARM	Z_	ZSIGN	ZOPTN	ZLOW	ZHIGH	ZNOTA	ZUTIL
ACTIVATION		I	EQ			ACTIVATION AUTOMATIC POST	X
INVE_THRESHOLD_BIGP		I	EQ	3		THRESHOLD FOR PHYSICAL INVENTORY DOCUMENT FOR ST BIG	X
INVE_THRESHOLD_FP00		I	EQ	3		THRESHOLD FOR PHYSICAL INVENTORY DOCUMENT FOR ST FP00	X
INVE_THRESHOLD_MDBX		I	EQ	5		THRESHOLD FOR PHYSICAL INVENTORY DOCUMENT FOR ST MDBX	X
INVE_THRESHOLD_SMBX		I	EQ	5		THRESHOLD FOR PHYSICAL INVENTORY DOCUMENT FOR ST SMBX	X
INVE_THRESHOLD_SMSU		I	EQ	10		THRESHOLD FOR PHYSICAL INVENTORY DOCUMENT FOR ST SMSU	X
INVE_THRESHOLD_SWU0		I	EQ	3		THRESHOLD FOR PHYSICAL INVENTORY DOCUMENT FOR ST SWU0	X

Figura 4.14: Tuple inserite per il posting automatico dei documenti

Tabella ZSPC_PHY_DOC

Nel caso in cui la differenza tra le quantità registrate all'interno del documento di inventario e le rimanenze di magazzino presenti nel sistema informativo superi la soglia percentuale stabilita per la categoria di stoccaggio di appartenenza, il programma deve provvedere ad inserire, dopo aver impostato lo stato del documento come *DELE*, le informazioni in esso contenuto all'interno di una tabella personalizzata. Questa tabella viene poi utilizzata da un Report appositamente realizzato per mostrare al responsabile i documenti che non hanno superato il controllo e permettergli di revisionarli manualmente o, in casi prestabiliti, provvedere alla generazione automatica di nuovi documenti di inventario per una nuova conta da parte degli operatori.

La tabella presenta la struttura visibile in figura 4.15.

Sviluppo del Function Module ZSPC_RF_SAVE_ZIVCOU

Per illustrare il processo funzionale sviluppato per implementare le funzionalità di conferma automatica dei documenti di inventario si prende in esame lo *Step IVMHU*. Le modifiche al salvataggio dei documenti sono state apportate nella stessa misura negli altri *Step* citati nella sezione 4.3.1 .

Al termine della procedura di conta inventariale per tutte le *HU* presenti

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF PERSONALIZZATA

Field	Key	Inl...	Data element	Dat...	Length	De...	Short Description
<u>LGNUM</u>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/SCWM/LGNUM	CHAR	4	0	Warehouse Number/Warehouse Complex
<u>DOC NUMBER</u>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/LIME/PI_DOC NU..	NUMC	20	0	Number of Physical Inventory Document
<u>ITEM NO</u>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/LIME/LINE_ITEM..	NUMC	6	0	Item
<u>DOC YEAR</u>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/LIME/PI_DOC YE..	NUMC	4	0	Document Year of Physical Inventory Document
<u>PROCESS TYPE</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/PI_PROCES..	CHAR	4	0	Process Type
<u>PI AREAD</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/PI_DE_PI..	CHAR	20	0	Physical Inventory Area
<u>DOC TYPE</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/PI_DOCUME..	CHAR	2	0	Physical Inventory Procedure (Document Type of Phys. Inv..
<u>DOC STATUS</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/PI_STATUS	CHAR	4	0	Physical Inventory Status
<u>COUNT DATE</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/PI_COUNT..	DEC	21	7	Entered Date of Physical Inventory Count (Time Stamp)
<u>COUNT USER</u>	<input type="checkbox"/>	<input type="checkbox"/>	UNAME	CHAR	12	0	User Name
<u>CREATE DATE</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/PI_CREATE..	DEC	21	7	Creation Date of Physical Inventory Doc. Item (Time Stamp)
<u>BOOK QUANTITY</u>	<input type="checkbox"/>	<input type="checkbox"/>	ZSPCFI_BOOK_QUA..	QUAN	31	14	Book quantity in inventory management
<u>COUNTED QUANTITY</u>	<input type="checkbox"/>	<input type="checkbox"/>	ZSPCFI_COUNTED..	QUAN	31	14	Counted quantity in inventory management
<u>DIFF QUANTITY</u>	<input type="checkbox"/>	<input type="checkbox"/>	ZSPCFI_DIFF_QUA..	QUAN	31	14	Diff quantity in inventory management
<u>THRESHOLD QTY</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/PI_QUANTI..	QUAN	31	14	Quantity in Inventory-Managed Unit of Measure
<u>DOC YEAR REF</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/PI_DOC YE..	NUMC	4	0	Document Year of Physical Inventory Document
<u>DOC REF</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/PI_DOC NU..	NUMC	20	0	Number of Physical Inventory Document
<u>ITEM REF</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/LINE_ITEM..	NUMC	6	0	Item
<u>PRODUCT</u>	<input type="checkbox"/>	<input type="checkbox"/>	/SCWM/DE_MATNR	CHAR	40	0	Product
<u>LGFLA</u>	<input type="checkbox"/>	<input type="checkbox"/>	/SCWM/LGFLA	CHAR	18	0	Storage Bin
<u>WHO</u>	<input type="checkbox"/>	<input type="checkbox"/>	/LIME/REF_DOC_ID	CHAR	70	0	Document ID of Reference Document
<u>STATUS_ICON</u>	<input type="checkbox"/>	<input type="checkbox"/>	CHAR4	CHAR	4	0	Not More Closely Defined Area, Possibly Used for Patchlevels
<u>TIMLO CREATE</u>	<input type="checkbox"/>	<input type="checkbox"/>	TIMS	TIMS	6	0	Field of type TIMS
<u>TIMLO COUNT</u>	<input type="checkbox"/>	<input type="checkbox"/>	TIMS	TIMS	6	0	Field of type TIMS
<u>DATLO COUNT</u>	<input type="checkbox"/>	<input type="checkbox"/>	DATS	DATS	8	0	Field of type DATS
<u>DATLO CREATE</u>	<input type="checkbox"/>	<input type="checkbox"/>	DATS	DATS	8	0	Field of type DATS
<u>STATUS_CR_COUNT</u>	<input type="checkbox"/>	<input type="checkbox"/>	ZSPCFI_STATUS_C..	STRI..	0	0	Status count for Physical Inventory Document

Figura 4.15: Struttura della tabella ZSPC_PHY_DOC

nell'unità di stoccaggio è necessario salvare i dati registrati all'interno del documento di inventario che si sta elaborando. Il salvataggio di queste informazioni è legato all'attivazione del *Function Code* **SAVE**, associato di default alla pressione del pulsante **F11** da parte dell'operatore. All'interno del *Function Module* *PAI* dello *Step* in questione è presente la logica applicativa per la gestione di ogni *Function Code* associato alla transazione; il processo standard prevede che, dopo aver effettuato una verifica sui dati inseriti in memoria, questi vengano elaborati ed inseriti all'interno del documento di inventario e che questo sia aggiornato permanentemente all'interno del database, modificando la proprietà contenente lo stato con il valore **COUNT**.

L'implementazione della nuova funzionalità di conferma automatica del documento è stata integrata in questa sezione del flusso di elaborazione, inserendo una chiamata ad un *Function Module* esterno al cui interno è stata inserita la logica di business per il controllo e l'approvazione automatica del documento, come mostrato di seguito.

```
1 " Recupero i dati dello Storage Bin analizzato
2 CLEAR: lv_lgpla.
3 MOVE ivhead-lgpla_verif TO lv_lgpla.
4
5 " Invocazione Function Module per il Posting
   automatico
6 REFRESH: lt_bapiret.
7 CLEAR l_mtype.
8 CALL FUNCTION 'ZSPC_RF_SAVE_ZIVCOU'
9   EXPORTING
10     iv_lgpla      = lv_lgpla
11     is_item_head  = ls_item_head
12     iv_matnr     = ls_count-matnr
13     ivhead       = ivhead
14   IMPORTING
15     et_bapiret   = lt_bapiret
16     e_rc_severity = l_mtype.
17
18 " Utility di sistema per il controllo degli errori
19 PERFORM error_analysis
20   USING
21     l_mtype
22     lt_bapiret.
```

Listing 4.3: Invocazione del Function Module per il Posting automatico

Il *Function Module* personalizzato riceve in ingresso i seguenti parametri:

- **lv_lgpla:** struttura dati contenente le informazioni sullo **Storage Bin** oggetto della conta. Il campo *LGPLA* è largamente utilizzato all'interno del mondo SAP EWM.
- **ls_item_head:** struttura che mantiene tutti i dati dell'intestazione del documento di inventario, tra cui la sua tipologia ed il numero progressivo di identificazione.
- **ls_count-matnr:** codice materiale del prodotto che si sta contando, utilizzato per il salvataggio del riferimento all'interno della tabella custom *ZSPC_PHY_DOC*.

- **ivhead**: documento di testa del processo di inventario fisico, mantiene il numero di *Warehouse Order* dell'operazione corrente.

Si analizza ora l'implementazione del modulo, contenuto all'interno del pacchetto *ZSPC_MANF* e denominato **ZSPC_RF_SAVE_ZIVCOU**, utilizzato per le implementazioni di questo progetto. La scelta di separare il codice funzionale dal modulo *PAI* degli *Step* è stata presa per ridurre la ridondanza degli applicativi ed unificare la logica in un unico file.

Si vogliono evidenziare le seguenti sezioni del modulo funzionale:

- Una volta recuperati i dati dell'unità di stoccaggio in esame, è necessario estrarre dalla tabella di configurazione *ZSPC_PARM* la soglia di interesse per la valutazione del documento di inventario. Come visto precedentemente, la soglia dipende dal parametro **Storage Type** dell'ubicazione selezionata. Questo dato viene concatenato con la stringa costante arbitraria *INVE_THRESHOLD_* a formare il nome del parametro da ricercare all'interno della tabella.

```
1  " Costruisco il nome del parametro contenente le
   informazioni
2  CLEAR lv_zparm_name.
3  CONCATENATE lc_zparm_threshold lv_lgtyp INTO
   lv_zparm_name.
4
5  " Recupero la tupla contenente la soglia di
   interesse
6  SELECT SINGLE *
7  FROM zspc_parm
8  INTO ls_zspc_parm
9  WHERE
10     zproc = gv_lgnum AND
11     repid = 'ZSPC_INVE_AUTO_POST' AND
12     zparm = lv_zparm_name.
```

Listing 4.4: Recupero del parametro soglia dalla tabella personalizzata

- Una volta recuperati l'oggetto di monitoraggio delle rimanenze a sistema ed il documento di conta inventariale devo estrarre le quantità

di materiale da confrontare per verificare il superamento della soglia prestabilita. Ottenute le quantità posso calcolare il valore di soglia in percentuale sulle rimanenze a sistema e, quindi, calcolare le differenze per stabilire se la soglia è stata superata oppure il documento può essere approvato senza ulteriori verifiche. Da notare che all'interno del documento di inventario, una volta effettuati i controlli al termine della conta, sono già presenti le differenze con lo stock presente nel sistema informativo, evitando quindi che queste debbano essere calcolate esplicitamente.

```
1  " Calcolo la soglia a partire dalla percentuale
2  lv_threshold = lv_book_quan / 100 * ls_zspsc_parm -
      zlow.
3
4  " Calcolo la differenza totale, contando i segni
5  CLEAR: lv_total_diff, ls_diff, ls_quan_temp.
6  LOOP AT ls_item_read-t_difference INTO ls_diff.
7    READ TABLE ls_diff-t_quan INTO ls_quan_temp INDEX
      1.
8    IF ls_diff-data-dif_direction EQ '0'.
9      lv_total_diff = lv_total_diff - ls_quan_temp -
      quantity.
10   ELSE.
11     lv_total_diff = lv_total_diff + ls_quan_temp -
      quantity.
12   ENDIF.
13 ENDLOOP.
14
15 IF lv_total_diff GE lv_threshold.
16   lv_over = abap_true.
17 ENDIF.
```

Listing 4.5: Verifica del superamento della soglia prestabilita

- In caso l'elaborazione verifichi che la soglia è stata superata il *Function Module* provvede alla cancellazione del documento di inventario ed inserisce le informazioni di interesse all'interno della tabella

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF PERSONALIZZATA

ZSPC_PHY_DOC. Da notare come, una volta chiamato il *Function Module* di cancellazione del documento, viene esplicitamente invocata la *Commit* della modifica in modo che il sistema registri le modifiche sul database prima che l'elaborazione prosegua.

```
1 REFRESH: et_bapiret, lt_item_to_read.
2 CLEAR e_rc_severity.
3 CALL FUNCTION '/SCWM/PI_CALL_DOCUMENT_DELETE'
4   EXPORTING
5     is_head      = ls_head_attr
6     it_item      = lt_item_to_delete
7   IMPORTING
8     et_pi_doc    = lt_item_to_read
9     et_bapiret   = et_bapiret
10    e_rc_severity = e_rc_severity.
11
12 IF NOT l_rc_severity CA wmegc_severity_ea.
13   COMMIT WORK AND WAIT.
14
15 [...]
16
17 "Aggiunta dati alla tabella custom ZSPC_PHY_DOC
18 CLEAR ls_zspc_phy_doc.
19 DATA(item_data) = ls_item_read-data.
20 MOVE-CORRESPONDING item_data TO ls_zspc_phy_doc.
21 ls_zspc_phy_doc-pi_ared = lv_pi_ared.
22 ls_zspc_phy_doc-book_quantity = lv_book_quan.
23 ls_zspc_phy_doc-counted_quantity = lv_count_res.
24 ls_zspc_phy_doc-diff_quantity = lv_total_diff.
25 ls_zspc_phy_doc-threshold_qty = lv_threshold.
26
27 " Devo controllare se ci sono riferimenti
   inseriti nel documento
28 CLEAR s_logitem.
29 LOOP AT ls_item_read-t_logitem INTO s_logitem.
30   IF s_logitem-ref_doc_type EQ 'SCWM-RFUI'.
31     SPLIT s_logitem-ref_doc_id AT '-'
```

```
32         INTO ls_zspc_phy_doc - doc_year_ref
           ls_zspc_phy_doc - doc_ref ls_zspc_phy_doc
           - item_ref .
33     ENDIF .
34 ENDLOOP .
35
36 ls_zspc_phy_doc - product = iv_matnr .
37 ls_zspc_phy_doc - lgpla = iv_lgpla .
38 CONCATENATE ls_item_read - data - lgnum ivhead - whord
           INTO ls_zspc_phy_doc - who .
39
40 INSERT zspc_phy_doc FROM ls_zspc_phy_doc .
41 ENDIF .
```

Listing 4.6: Cancellazione del documento e copia delle informazioni

L'ultima cosa che si vuole evidenziare è lo script per il controllo di eventuali riferimenti presenti nel documento. Come si vedrà nel capitolo successivo del presente elaborato, il programma di Report per la gestione dei documenti bloccati in fase di conferma permette al responsabile di creare automaticamente nuovi documenti di inventario a partire da quelli eliminati. In questo caso, però, all'interno del documento generato viene inserito, nel campo **logitem**, il riferimento al documento a partire dal quale è avvenuta la creazione. Questo riferimento viene poi copiato nuovamente nella tabella nel caso in cui anche la nuova conta dovesse fallire; in questo modo viene tenuta traccia dell'iter complessivo.

4.4.3 Esecuzione della transazione logica

L'intero codice oggetto del presente capitolo viene reso disponibile all'interno dell'appendice dell'elaborato di tesi. In questa ultima sezione del capitolo si vuole, invece, mostrare l'intera esecuzione della transazione logica sviluppata in modo da evidenziarne il processo funzionale.

Una volta mandata in esecuzione la transazione */SCWM/RFUI* il sistema propone una semplice interfaccia mediante la quale l'operatore può effettuare l'accesso alla propria area riservata. L'ambiente operativo di ogni utente del sistema è individuato mediante 3 campi: l'identificativo del magazzino di appartenenza, la risorsa che si sta adoperando ed il *Personalization Pro-*

file che si vuole utilizzare. Ricordiamo che quest'ultimo determina anche il menu di transazioni accessibili all'operatore.

Una volta effettuato l'accesso è necessario navigare all'interno del menu proposto per individuare ed avviare la transazione in radiofrequenze che si vuole utilizzare. In figura 4.16 viene mostrato il percorso da effettuare per selezionare la transazione di inventario personalizzata, in particolare la terza voce presente nella figura 4.16c.

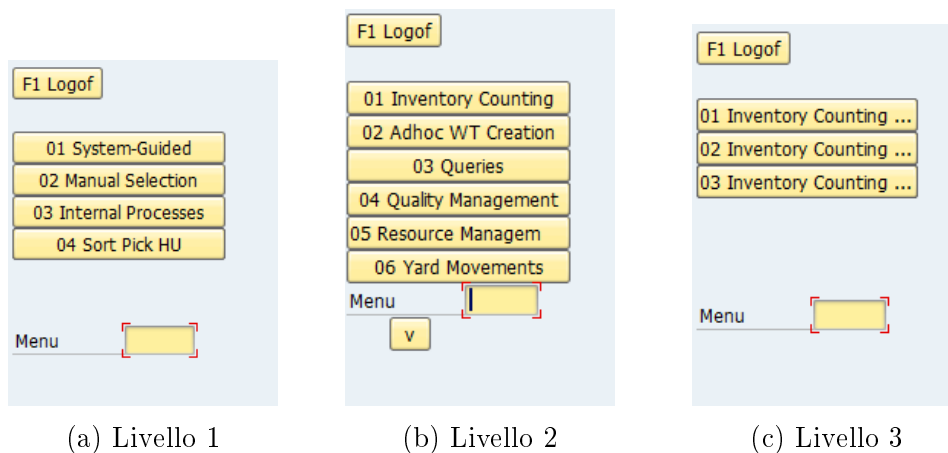


Figura 4.16: Percorso menu per la transazione ZIVCOU

A questo punto il sistema informativo individua eventuali task di magazzino, relativi a documenti di inventario in attesa, assegnati all'operatore. In caso di successo viene selezionato il primo documento e viene proposta a schermo una *Storage Bin* da controllare, come mostrato in figura 4.17. L'operatore deve recarsi fisicamente presso l'unità e scansionare all'interno del campo di input in codice identificativo della *Bin*; alla pressione del tasto *ENTER* il sistema ne verifica la corrispondenza e naviga verso la schermata successiva.

Questa è la finestra di inserimento delle *Handling Unit* presenti fisicamente all'interno dell'ubicazione. Per ognuna di queste, l'utente deve scansionarne l'etichetta e premere il tasto *ENTER* per aggiungerla alla lista delle unità da contare. Nel caso in cui non fossero presenti *HU*, poiché la *Bin* è configurata per ospitare prodotti sfusi, basterà premere il tasto funzione *F2* per aprire la relativa schermata in cui inserire il numero di pezzi contati. Si vuole far notare come il sistema supporta fino a due livelli innestati: è

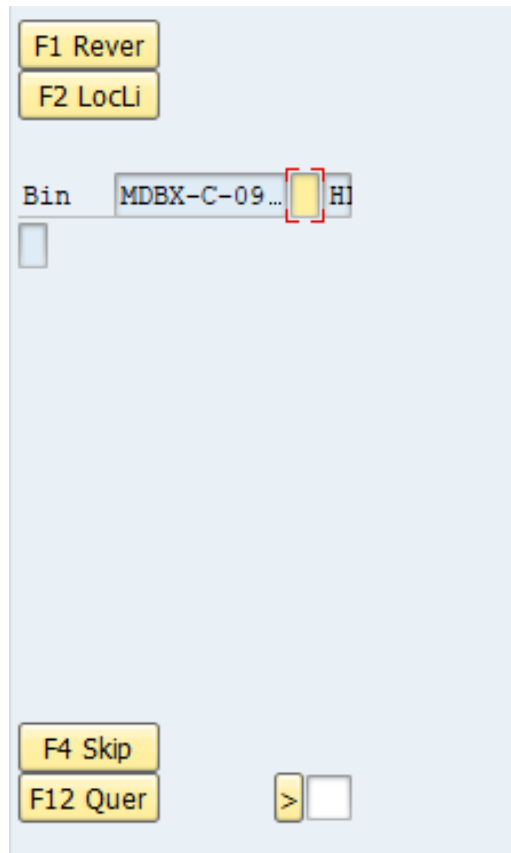


Figura 4.17: Scansione della Bin da verificare

possibile che un'unità di stoccaggio mantenga i beni all'interno di una *HU* a sua volta contenuta in un altro contenitore. Nel caso specifico di questa transazione i prodotti vengono riposti all'interno di una sola *HU*. In figura 4.18 viene mostrata la scansione di una di queste e la sua aggiunta alla lista; premendo un'ulteriore volta il tasto *ENTER* viene effettuata la navigazione verso la relativa schermata di gestione. Nella finestra proposta di seguito all'operatore è necessario utilizzare il tasto **F2** per inserire i dati relativi al prodotto stoccato nel contenitore selezionato. Il sistema mostra, quindi, la schermata personalizzata *9999*, la cui implementazione è stata analizzata in questo capitolo. In figura 4.19 è possibile notare come sia richiesta la scansione del codice prodotto qui presente; una volta inserito il dato all'interno del capo selezionato e premuto il tasto *ENTER* viene richiamato il modulo

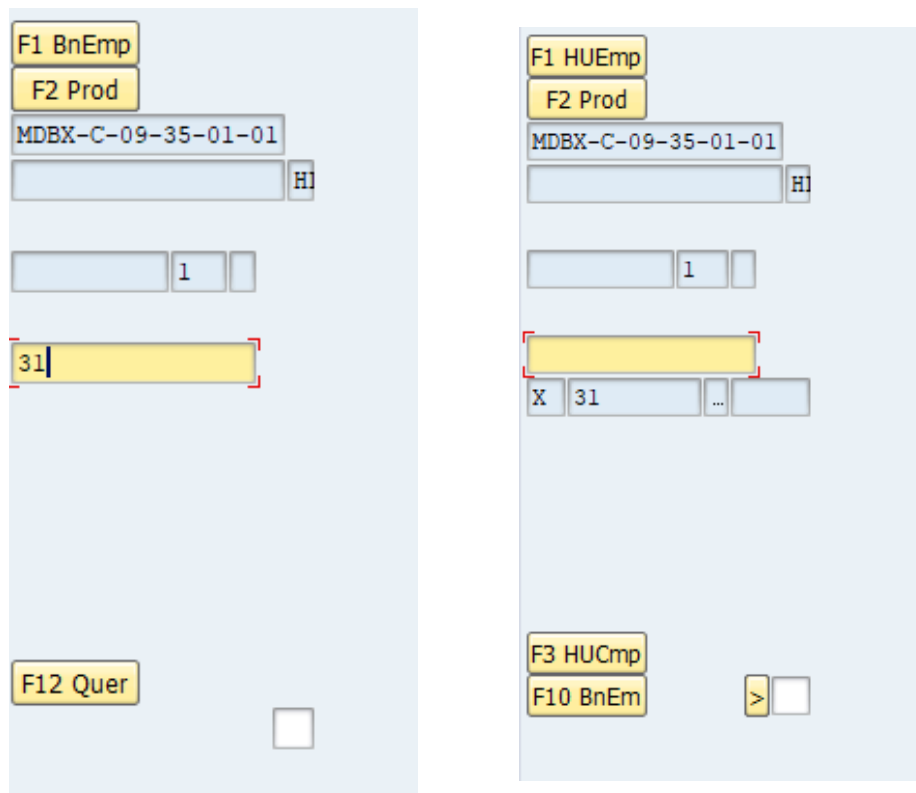


Figura 4.18: Scansione di una HU all'interno dell'ubicazione

funzionale per la compilazione automatica di alcuni dei campi a schermo. A questo punto è sufficiente inserire il numero di pezzi fisicamente presenti e confermare nuovamente con il tasto *ENTER*.

Nell'ambiente RF alcuni tasti funzione sono associati a dei *Function Code* standard. Fino ad ora abbiamo avuto come esempio il tasto *ENTER*, utilizzato per confermare gli input e navigare verso la schermata immediatamente successiva nel flusso della transazione logica. Un altro dei tasti associati ad operazioni di routine è **F7**, utilizzato per tornare alla schermata precedente. Terminato l'inserimento del numero di pezzi presenti in ogni *HU* dell'ubicazione l'operatore deve confermare il salvataggio dei dati all'interno del documento di inventario collegato. Questa operazione viene effettuata premendo il tasto funzione **F11** all'interno della pagina di riepilogo delle *HU* contate. Il *Function Code* associato inserisce i dati nel documento e succes-

CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF PERSONALIZZATA

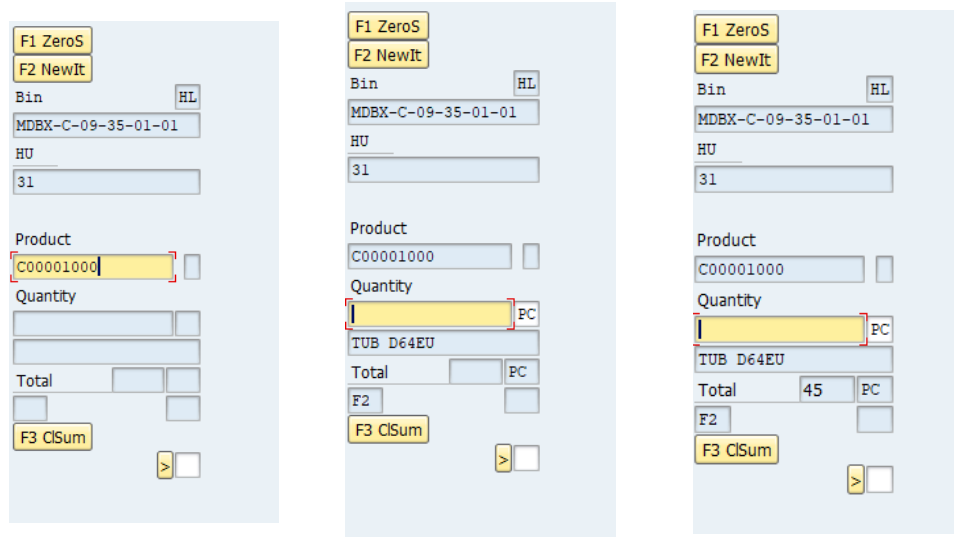


Figura 4.19: Processo di conteggio dei prodotti presenti nel contenitore

sivamente invoca il *Function Module* **ZSPC_RF_SAVE_ZIVCOU** da noi implementato. Dipendentemente dal risultato della verifica il documento di inventario viene confermato oppure il suo stato viene impostato come *DELE* nel caso in cui le differenze rilevate siano troppo marcate. In questo secondo caso, inoltre, i dati del documento vengono inseriti all'interno della tabella *ZSPC_PHY_DOC* per essere revisionati in seguito.

*CAPITOLO 4. IMPLEMENTAZIONE DELLA TRANSAZIONE RF
PERSONALIZZATA*

Capitolo 5

Realizzazione del Report di revisione dei documenti di inventario

Una volta sviluppata nella sua interezza la modifica alla transazione logica di conteggio di un documento di inventario si è provveduto a soddisfare le richieste dell'ultimo punto evidenziato nella soluzione esposta nel capitolo 3. Per dare ai revisori uno strumento in grado di valutare i documenti di inventario bloccati in fase di conferma ed, eventualmente, crearne di nuovi si è scelto di realizzare un programma *Dynpro* standard composto da due elementi: il primo è una visualizzazione tabellare dei dati memorizzati nella tabella personalizzata *ZSPC_PHY_DOC*, al cui interno vengono salvati le informazioni di interesse dei documenti la cui verifica è fallita; il secondo, invece, è la possibilità di creare un nuovo documento a partire da una tupla della tabella mediante la pressione di un singolo tasto. Per fare ciò, è stato necessario creare la schermata principale del programma ed, in seguito, integrare la logica funzionale per popolare la vista dinamicamente con i dati presenti nel database e permettere la gestione di nuovi documenti mediante il tasto preposto inserito all'interno dell'interfaccia grafica.

Nella prima sezione verrà mostrato il processo di creazione del programma e di definizione della schermata principale, lasciando alla seconda il compito di illustrare in dettaglio lo sviluppo effettuato.

5.1 Definizione dell'interfaccia

Sia la creazione del programma che la definizione dei suoi *Function Module* e delle sue schermate è stata eseguita all'interno della transazione **Object Navigato (SE80)**, adoperando gli strumenti messi a disposizione dalla piattaforma per la gestione del Report nella sua interezza.

In particolare, il primo passo è stato quello di creazione del programma all'interno del *Package* definito nelle prime fasi progettuali. Una volta aperto il contenitore all'interno della transazione possiamo notare le seguenti sotto-cartelle, fra le altre:

- **Dictionary Objects:** oggetti come strutture e tabelle, creati mediante la transazione *SE11* ed utilizzati sia per il mantenimento permanente di dati informativi all'interno del sistema che come elementi di configurazione a supporto dei programmi ABAP.
- **Class Library:** contiene tutti gli elementi del pacchetto generati secondo il paradigma ad oggetti, come classi ed interfacce.
- **Programs:** insieme dei programmi ABAP definiti all'interno del *Package*.
- **Function Groups:** insieme dei *Function Module*, raggruppati all'interno di gruppi per una migliore organizzazione dei sorgenti.
- **Includes:** file di inclusione globali i cui riferimenti vengono utilizzati dai programmi del contenitore.

Nel nostro caso, la creazione del nuovo Report è resa agevole dalla possibilità di svolgere l'operazione mediante il click destro sulla directory *Programs*, scegliendo la creazione di un nuovo programma. Il **Dialog** di configurazione viene mostrato in figura 5.1. All'interno di questa schermata è possibile configurare i parametri che determinano il comportamento del programma e la sua interazione con l'ambiente di esecuzione. In dettaglio, il Report è stato impostato come programma eseguibile di tipo personalizzato ed è stato scelto come ambiente di esecuzione *SAP Basis*. Il nome del programma è stato definito per rispecchiare pienamente, come buona pratica prevede, lo scopo e le funzionalità dello stesso.

Successivamente è stata sviluppata la prima ed unica **Screen** del programma, essendo tutte le funzionalità utilizzabili in un'unica finestra. Sempre

CAPITOLO 5. REALIZZAZIONE DEL REPORT DI REVISIONE DEI DOCUMENTI DI INVENTARIO

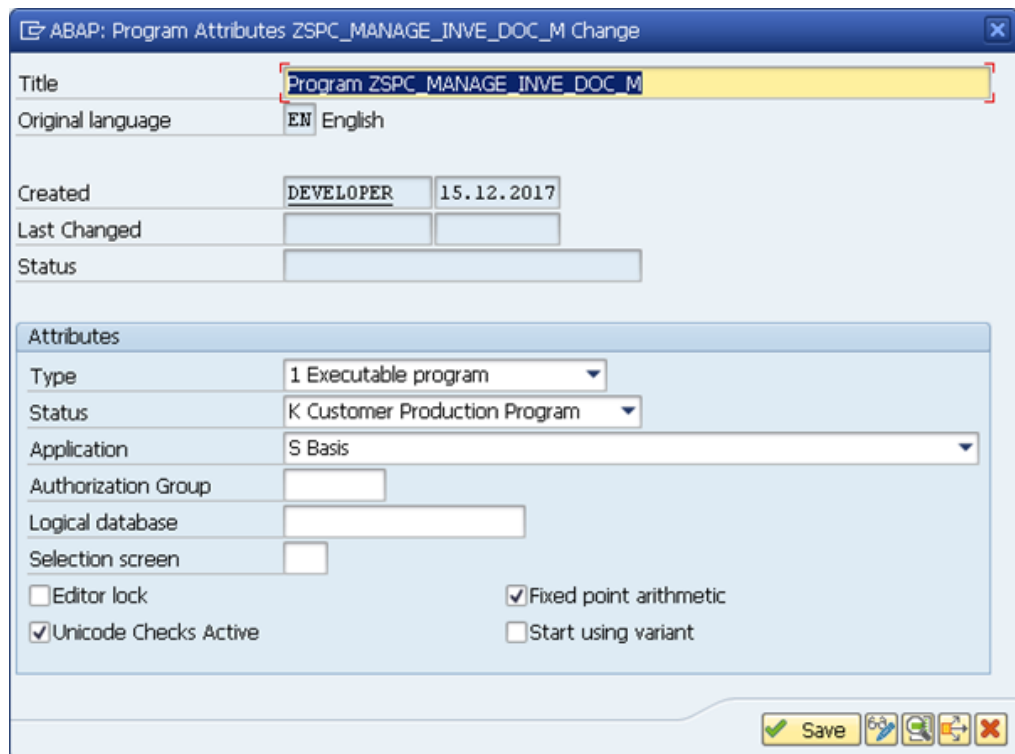


Figura 5.1: Dialog di configurazione del programma

utilizzando lo strumento *Object Navigator* si è aperta la sezione relativa all'applicativo appena definito. Da qui, mediante l'utilizzo del menu contestuale accessibile mediante un click destro sulla directory **Screens** è stata avviata la procedura di creazione di una nuova schermata scegliendo come numero identificativo **9999**. In figura 5.2 vengono mostrati i dettagli implementativi della schermata. L'unico elemento inserito all'interno della finestra è stato un *Container* generico al cui interno, durante l'esecuzione, viene elaborato e renderizzato un oggetto **ALV GRID**.

ALV Grid Control è uno strumento, messo a disposizione dal framework di sviluppo ABAP e sviluppato utilizzando il paradigma ad oggetti, tramite il quale renderizzare e gestire liste di elementi. Viene solitamente utilizzato in programmi ABAP per visualizzare il contenuto di una tabella di sistema, aggiungendo o rimuovendo campi e mettendo a disposizione dell'utente un insieme di funzioni generiche per manipolare i dati mostrati quali, ad esem-

Short Description	Main screen		
Original Language	EN	English	Package ZSPC_00
Last Changed	19.12.2017	10:27:44	
Last Generation	19.12.2017	10:27:49	

Screen Type	Settings
<input checked="" type="radio"/> Normal <input type="radio"/> Subscreen <input type="radio"/> Modal dialog box <input type="radio"/> Selection screen	<input type="checkbox"/> Hold Data <input type="checkbox"/> Switch Off Runtime Compress <input type="checkbox"/> Template - non-executable <input type="checkbox"/> Hold Scroll Position <input type="checkbox"/> Without Application Toolbar

Other Attributes			
Next Screen		9999	
Cursor Position			
Screen Group			
Lines/Columns	Occupied	27	175
	Mainten.	27	176
Context Menu FORM ON CTMENU			<input type="button" value="Properties"/>

Figura 5.2: Finestra di creazione di una nuova schermata

pio, strumenti di selezione e filtraggio. Il componente grafico è composto da 3 elementi separati: una barra degli strumenti, un titolo e la tabella contenente i dati; a discrezione dello sviluppatore è possibile anche nascondere i primi due.

Il componente viene creato e popolato in fase di esecuzione da parte del *Function Module PBO* eseguito prima di renderizzare la finestra. Ad ogni *Screen*, infatti, è possibile associare un numero arbitrario di moduli funzionali per gestirne il comportamento; i moduli presenti di default permettono la gestione delle sequenze di visualizzazione delle schermate all'interno di un programma. In questo caso sono stati realizzati due moduli, parte del file **ZSPC_MANAGE_INVE_DOC_M_F01**, al cui interno sono presenti anche diverse *Subroutine* utilizzate dal programma. Tutto il codice è visionabile in appendice al presente lavoro di tesi. Di seguito viene mostrato un estratto di codice per mostrare il processo di creazione e configurazione

CAPITOLO 5. REALIZZAZIONE DEL REPORT DI REVISIONE DEI DOCUMENTI DI INVENTARIO

del componente grafico. Questo viene creato passando in ingresso la tabella interna globale *gt_zspc_phy_doc* definita all'interno del codice del Report prima che venga richiamata la schermata 9999.

```
1  " Creazione dell'ALV con cattura delle eccezioni
2  TRY.
3      cl_salv_table=>factory(
4          EXPORTING
5              r_container = gr_container
6              container_name = 'ALV_CONTAINER'
7          IMPORTING r_salv_table = gr_alv
8          CHANGING t_table = gt_zspc_phy_doc ).
9      CATCH cx_salv_msg .
10 ENDTRY.
11
12 ...
13
14 " Rendo visibili tutte le funzioni dell'ALV
15 gr_functions = gr_alv->get_functions( ).
16 gr_functions->set_all( if_salv_c_bool_sap=>true ).
17 " Aggiungo il pulsante per richiamare la funzione
    custom di creazione documenti
18 TRY .
19     gr_functions->add_function(
20         EXPORTING
21             name = 'FUNC_CREA_DOC'
22             text = 'Create Inventory Document'
23             tooltip = 'Create Inventory Document'
24             position = if_salv_c_function_position=>
                right_of_salv_functions ).
25     CATCH cx_salv_existing cx_salv_wrong_call.
26 ENDTRY.
27 " Impostazione del componente, tra cui il titolo
28 CLEAR gr_display.
29 MOVE 'SPC: Physical inventory monitor' TO lv_title.
30 gr_display = gr_alv->get_display_settings( ).
31 gr_display->set_list_header( lv_title ).
32
```

```

33 ...
34
35 " Alcune colonne della tabella vengono nascoste
36 CLEAR gr_column .
37 gr_column = gr_columns->get_column( 'CREATE_DATE' ).
38 gr_column->set_visible( if_salv_c_bool_sap=>false ).
39 CLEAR gr_column .
40 gr_column = gr_columns->get_column( 'COUNT_DATE' ).
41 gr_column->set_visible( if_salv_c_bool_sap=>false ).
42
43 " Renderizzazione del componente grafico
44 gr_alv->display( ).

```

Listing 5.1: Creazione dell'istanza ALV Grid Control

Come è possibile notare una delle funzionalità offerte da *ALV* è quella di integrare all'interno della barra degli strumenti anche pulsanti personalizzati e collegati a funzioni create ad hoc. In questo caso è stato aggiunto il pulsante per la creazione di un nuovo documento di inventario collegandolo alla funzione `create_new_document` discussa nella sezione successiva. In figura 5.3 viene mostrato il risultato di quanto implementato.

Whse No.	PI Doc.	Item	Doc.	Year	Proc.	Type	PI Area	Procedure	Status	Count date	Count time	User Name	Creation date	Crea. time	Book Quantity	Counted Quantity	Diff Quantity
SCAR	40	1	2017	SCWM	I08C	HL	DELE	14.12.2017	09:55:42	DEVELOPER	14.12.2017	09:54:52	100,00000000000000	110,00000000000000	10,00000000000000		
SCAR	37	1	2017	SCWM	I08C	HL	DELE	13.12.2017	17:02:16	DEVELOPER	13.12.2017	16:59:43	0,00000000000000	0,00000000000000	0,00000000000000		
SCAR	51	1	2017	SCWM	I08C	HL	DELE	18.12.2017	15:43:25	DEVELOPER	18.12.2017	15:42:20	101,00000000000000	135,00000000000000	34,00000000000000		

Figura 5.3: Tabella ZSPC_PHY_DOC visualizzata mediante ALV Grid

5.2 Implementazione delle funzionalità

Questa sezione ha l'obiettivo di analizzare due funzionalità implementate all'interno del programma. La prima è il recupero dei dati contenuti all'interno della tabella custom *ZSPC_PHY_DOC* e la loro elaborazione atta a ricavare informazioni complesse sullo stato del sistema; successivamente, invece, si va a commentare la funzionalità di creazione di un nuovo documento di inventario offerta nel caso in cui vengano rispettate determinate

condizioni.

Innanzitutto, il programma scarta le tuple relative a documenti che presentano lo stato *ACTI* o *POST* in quanto sono documenti i cui riferimenti sono stati inseriti erroneamente nella tabella - solo i documenti in stato **DELE** dovrebbero essere presenti. Inoltre, vengono anche scartati i documenti di tipo *HS*, ovvero di conta inventariale per prodotto e non per unità di stoccaggio. Qualora un documento di questo tipo venisse bloccato durante la verifica non dovrebbe essere possibile crearne uno nuovo previa approvazione manuale da parte del supervisore.

Successivamente viene processato il contenuto dei campi ricavati a tempo d'esecuzione e non presenti all'interno della tabella salvata nel database. Questi campi, aggiunti alla struttura tabellare renderizzata dal componente *ALV*, sono i seguenti:

- *Stato della tupla* ogni riga della tabella può avere uno dei seguenti 3 stati: **FIRST** in caso sia un documento *HL* senza alcun riferimento, quindi al primo conteggio fallito; **SECOND** se è un documento *HL* che presenta un riferimento ad un altro documento, in questo caso ha dato esito negativo anche la seconda operazione di conta; **THIRD** se è riferita ad un documento *HS* che non ha, a sua volta, superato la verifica in fase di salvataggio.
- *Data e ora della creazione del documento.*
- *Data e ora della conta del documento.*

Mediante l'utilizzo della Subroutine **check_already_counting**, inoltre, viene verificato che non esista già un documento dello stesso tipo in elaborazione per lo stesso materiale o la stessa ubicazione. A seconda del risultato di questo controllo, il campo della tabella **status_icon** viene valorizzato con un'icona semaforica di colore verde o rosso, ad indicare visivamente la possibilità o meno di creare un nuovo documento. Una volta terminato anche questa verifica viene invocato, esplicitamente, il passaggio alla finestra di visualizzazione 9999.

Viene ora discussa l'implementazione della funzionalità di creazione di un nuovo documento. Alla pressione del pulsante preposto viene invocata la Subroutine - in quanto definita mediante la parola chiave *FORM* - **create_new_document**. Questa funzione prevede che l'utente abbia selezionato una riga della tabella, e solo una alla volta, ed in base al contenuto

di questa tupla stabilisce se è possibile creare un nuovo documento; in caso questo controllo risulti positivo, il documento creato dipende dalle informazioni selezionate.

Nel caso in cui l'icona di stato della tupla sia rossa - indicando quindi un altro documento inventariale appartenente alla stessa tipologia riferito allo stesso materiale o alla stessa ubicazione - la pressione del tasto non ha alcun effetto e viene mostrato un messaggio di errore. In caso positivo, invece, viene visualizzato un pop-up di conferma per verificare che l'utente voglia realmente generare un nuovo documento a partire da quella tupla, come mostrato in figura 5.4. Se l'utente ribadisce la sua intenzione allora la fun-

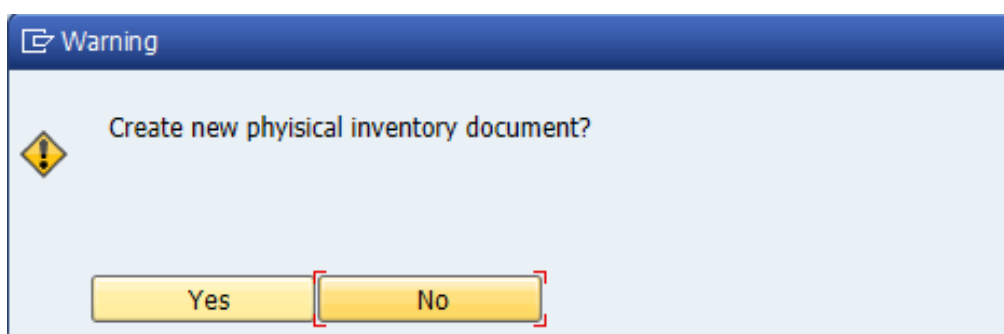


Figura 5.4: Popup di conferma alla creazione di un nuovo documento

zione effettua nuovamente la verifica riguardante un eventuale documento di conta già presente a sistema - nel tempo intercorso tra l'apertura della finestra e la pressione del tasto funzione potrebbe essere variato lo stato del sistema. Se il controllo ha nuovamente esito positivo, il metodo, a seconda dello stato del documento, eseguirà la seguente operazione:

- *FIRST*: Creazione di un documento *HL* focalizzato alla conta della stessa ubicazione di stoccaggio.
- *SECOND*: Creazione di un documento *HS* per la conta, in tutto il magazzino, delle quantità stoccate del prodotto che ha generato l'errore in fase di verifica.
- *THIRD*: Non accade nulla, il documento può solo essere confermato manualmente in quanto la conta ha dato esito negativo per 3 volte.

CAPITOLO 5. REALIZZAZIONE DEL REPORT DI REVISIONE DEI DOCUMENTI DI INVENTARIO

Di seguito viene mostrato, a scopo esemplificativo, il codice di creazione di un documento inventariale. La generazione avviene sfruttando un *Function Module* messo a disposizione da SAP come parte delle proprie librerie.

```
1 CLEAR ls_create_header.
2 ls_create_header-procedure = 'HL'.
3 ls_create_header-whsnumber = ls_zspsc_phy_doc-lgnum
4
5 CLEAR ls_create_item.
6 REFRESH lt_create_item.
7
8 CALL FUNCTION '/SCWM/BAPI_PI_DOCUMENT_CREATE'
9   EXPORTING
10    create_header      = ls_create_header
11   IMPORTING
12    et_keys            = lt_keys
13   TABLES
14    create_item        = lt_create_item
15    create_reference   = lt_create_reference
16    return              = lt_bapiret.
```

Listing 5.2: Creazione di un documento di inventario

Il codice di sviluppo può essere visualizzato nella sua interezza all'interno dell'appendice del presente elaborato.

*CAPITOLO 5. REALIZZAZIONE DEL REPORT DI REVISIONE DEI
DOCUMENTI DI INVENTARIO*

Conclusioni

Una volta concluso lo sviluppo della soluzione tutte le modifiche, opportunamente raccolte all'interno delle rispettive *CR*, dovranno essere trasportate prima sul sistema di test per essere collaudate e, successivamente in caso di comportamento conforme alle specifiche, sul sistema di produzione. In quest'ultimo, la nuova transazione verrà aggiunta ai processi operativi già esistenti così da poter essere utilizzato dagli utenti del sistema.

In questo caso, lo sviluppo della transazione logica in radiofrequenza analizzato in questo elaborato è stato limitato all'implementazione nel sistema di sviluppo. L'incarico di trasportare le modifiche nei successivi sistemi e di effettuare i test di collaudo è stato affidato ad un'altra figura aziendale. Secondo i dati raccolti, i test hanno verificato il corretto funzionamento della soluzione, confermando in larga parte le previsioni. In particolare, sono diminuiti drasticamente gli errori commessi dagli operatori durante le operazioni inventariali grazie, principalmente, ad un'interfaccia grafica più adatta alle dimensioni dei palmari utilizzati e alle semplificazioni implementate in fase di inserimento dei dati. Il processo è stato largamente automatizzato, prevedendo solo una minima interazione da parte degli utenti e limitando, così, i margini di errore.

Il nuovo processo di inventario fisico ha, inoltre, permesso di ridurre notevolmente i tempi di esecuzione dell'attività inventariale dell'intero magazzino; date le dimensioni dello stesso, capace allo stato attuale di gestire più di 90000 prodotti finiti, risulta chiaro come la minimizzazione del periodo di fermo dei processi dell'impianto sia cruciale ai fini del risparmio economico e dell'efficienza nella distribuzione. Si vuole ricordare, infatti, che il deposito gestisce lo stoccaggio e la spedizione di parti di ricambio per migliaia di prodotti differenti a livello europeo.

Analizzando, invece, il processo di sviluppo della soluzione e l'esperienza di utilizzo degli strumenti offerti da SAP e ABAP, si riportano di seguito

alcune impressioni.

Lo sviluppo in ambiente SAP richiede un lungo periodo di formazione volto ad acquisire una conoscenza generale del prodotto e dei suoi processi di funzionamento. L'ambiente è rigidamente strutturato e richiede allo sviluppatore l'apprendimento di tecniche di programmazione e di strumenti creati ad hoc per la gestione delle numerose funzionalità messe a disposizione. Durante il periodo di tirocinio effettuato presso *Engineering Ingegneria Informatico s.p.a* si è reso necessario un lungo periodo di studio dei manuali della piattaforma prima di poter iniziare ad implementare la soluzione; oltre all'apprendimento autonomo sono anche stati seguiti diversi corsi tenuti da altri dipendenti dell'azienda senza i quali non sarebbe stato possibile ottenere il risultato desiderato.

Appendice A

All'interno di questa appendice è stato inserito, in forma estesa, il codice della soluzione implementata in questo elaborato. Alcuni estratti di quanto mostrato in questa sezione sono stati ripresi nei precedenti capitoli.

ZSPC_RF_HU_IVHU_PBO_M

```
1 ...
2
3 " Importazione di variabili globali
4
5 CALL FUNCTION '/SCWM/RF_INV_GET_GLOBVAR'
6   IMPORTING
7     ev_lgnum = gv_lgnum
8     ev_who   = gv_who
9 *   EV_TIMEZONE           =
10 *   EV_GUIDED_INVENTORY  =
11 *   EV_INVENTORY_NOT_POST_ITEM      =
12 *   EV_IND_DOC_TYPE     =
13 *   EV_IND_DELIVERY    =
14 *   EV_WITH_HU         =
15     ev_level = gv_level
16 *   EV_CURRENT_LINE_HEAD      =
17 *   ES_STEP                   =
18 *   ES_COUNT_FOR_CW          =
19 *   ES_STOCK_FOR_CW          =
20 *   ES_QUAN_FOR_CW           =
21 *   ES_SN_FOR_CW             =
```

```
22 *   ET_HU_STAT           =
23 *   ET_BUNDLE=
24 *   ET_QUAN   =
25 *   ET_SN     =
26   eo_cust   = grefo_cust
27 *   EO_CUST_SCWM       =
28   eo_rfpi   = grefo_rfpi
29   eo_pack   = grefo_pack
30 *   EO_UI_FIELDS      =
31 *   EO_STOCK_ID       =
32 *   EO_PI_SELECTS    =
33 .
34
35 ...
```

ZSPC_RF_MAT_IVMHUV_PAI_M

```
1 ...
2
3 "Dichiarazioni Manfredi
4   DATA: lo_mon_stock      TYPE REF TO /scwm/
        cl_mon_stock ,
5 lt_matnr_rTYPE /scwm/tt_matnr_r ,
6 ls_matnr_rLIKE LINE OF lt_matnr_r ,
7 lr_matnr  TYPE rseloption ,
8 lrs_matnr LIKE LINE OF lr_matnr ,
9 lv_lgpla  TYPE /scwm/lgpla ,
10 lt_lgpla_rTYPE /scwm/tt_lgpla_r ,
11 ls_lgpla_rLIKE LINE OF lt_lgpla_r ,
12 lr_lgpla  TYPE rseloption ,
13 lrs_lgpla LIKE LINE OF lr_lgpla ,
14 lt_stock_overview TYPE /scwm/tt_inc_out_mon ,
15 ls_stock_overview LIKE LINE OF lt_stock_overview ,
16 lt_phy_stock      TYPE /scwm/tt_stock_mon ,
17 lv_error  TYPE xfeld .
18   DATA: lt_cat      TYPE /scwm/tt_cat_r ,
19 lt_owner   TYPE /scwm/tt_owner_r ,
20 lt_entitled TYPE /scwm/tt_entitled_r ,
21 lt_charg   TYPE /scwm/tt_charg_r ,
22 lt_lgber   TYPE /scwm/tt_lgber_r ,
23 lt_lgtyp   TYPE /scwm/tt_lptyp_r ,
24 lt_lptyp   TYPE /scwm/tt_lptyp_r ,
25 lt_aisle   TYPE /scwm/tt_aisle_r ,
26 lt_stack   TYPE /scwm/tt_stack_r ,
27 lt_level   TYPE /scwm/tt_level_r ,
28 lt_binsc   TYPE /scwm/tt_binsc_r ,
29 lt_depth   TYPE /scwm/tt_depth_r ,
30 lt_psa     TYPE /scwm/tt_psa_r ,
31 lt_rsrc    TYPE /scwm/tt_rsrc_r ,
32 lt_tuext   TYPE /scwm/tt_sel_tu_num_ext ,
33 lt_tsp     TYPE /scwm/tt_sel_tsp .
34
```

```

35 CALL FUNCTION '/SCWM/RF_INV_GET_GLOBVAR'
36 IMPORTING
37     ev_lgnum      = gv_lgnum
38     ev_who        = gv_who
39 *     EV_TIMEZONE  =
40     ev_guided_inventory = gv_guided_inventory
41 *     EV_INVENTORY_NOT_POST_ITEM      =
42 *     EV_IND_DOC_TYPE                  =
43 *     EV_IND_DELIVERY                  =
44 *     EV_WITH_HU                       =
45     ev_level      = gv_level
46 *     EV_CURRENT_LINE_HEAD            =
47 *     ES_STEP                          =
48 *     ES_COUNT_FOR_CW                  =
49 *     ES_STOCK_FOR_CW                  =
50 *     ES_QUAN_FOR_CW                   =
51 *     ES_SN_FOR_CW                     =
52 *     ET_HU_STAT                       =
53 *     ET_BUNDLE                        =
54 *     ET_QUAN                          =
55 *     ET_SN                            =
56     eo_cust       = grefo_cust
57 *     EO_CUST_SCWM=
58     eo_rfpi       = grefo_rfpi
59     eo_pack       = grefo_pack
60     eo_ui_fields = grefo_ui_fields
61     eo_stock_id  = go_stock_id
62 *     EO_PI_SELECTS =
63
64 ...
65
66 "Creo oggetto per il monitor stock
67 IF lo_mon_stock IS NOT BOUND.
68     CREATE OBJECT lo_mon_stock EXPORTING iv_lgnum =
        gv_lgnum.
69 ENDIF.
70

```

```

71 "Recupero la bin corrente dall'header del documento
    di inventario
72 CLEAR: ls_head, lv_lgpla.
73 READ TABLE ivheadtab INTO ls_head INDEX 1.
74 MOVE ls_head-lgpla_sel TO lv_lgpla.
75
76 REFRESH lt_matnr_r.
77 CLEAR ls_matnr_r.
78 ls_matnr_r-sign = wmegc_sign_inclusive.
79 ls_matnr_r-option = wmegc_option_eq.
80 ls_matnr_r-low = ls_count-matnr_verif.
81 ls_matnr_r-high = ls_count-matnr_verif.
82 APPEND ls_matnr_r TO lt_matnr_r.
83
84 REFRESH lt_lgpla_r.
85 CLEAR ls_lgpla_r.
86 ls_lgpla_r-sign = wmegc_sign_inclusive.
87 ls_lgpla_r-option = wmegc_option_eq.
88 ls_lgpla_r-low = lv_lgpla.
89 ls_lgpla_r-high = lv_lgpla.
90 APPEND ls_lgpla_r TO lt_lgpla_r.
91
92 "Recupero i dati dello stock dal monitor
93 CALL METHOD lo_mon_stock->get_stock_overview
94     EXPORTING
95         it_matnr_r = lt_matnr_r
96         it_cat_r   = lt_cat
97         it_owner_r = lt_owner
98         it_entitled_r = lt_entitled
99         it_charg_r = lt_charg
100        it_lgtyp_r = lt_lgtyp
101        it_lgber_r = lt_lgber
102        it_lgpla_r = lt_lgpla_r
103        it_lptyp_r = lt_lptyp
104        it_aisle_r = lt_aisle
105        it_stack_r = lt_stack
106        it_level_r = lt_level

```

```

107     it_binsc_r   = lt_binsc
108     it_depth_r   = lt_depth
109     it_psa_r     = lt_psa
110     it_rsrc_r    = lt_rsrc
111     it_tu_num_ext_r = lt_tuext
112     it_tsp_r     = lt_tsp
113     IMPORTING
114     et_stock_overview = lt_stock_overview
115     et_physical_stock = lt_phy_stock
116     ev_error         = lv_error.
117
118     READ TABLE lt_stock_overview INTO ls_stock_overview
119     INDEX 1.
120     IF ls_stock_overview IS NOT INITIAL.
121         IF ls_stock_overview-cat IS NOT INITIAL.
122             ls_count-cat = ls_stock_overview-cat.
123             ls_count-owner = ls_stock_overview-owner.
124             ls_count-entitled = ls_stock_overview-entitled.
125         ENDIF.
126     ENDIF.
127     IF ls_count IS NOT INITIAL.
128         MODIFY ivmattab FROM ls_count INDEX lv_line.
129     ENDIF.
130
131     ...
132
133     " Salvataggio del documento alla pressione del
134     "   tasto F11
135
136     CLEAR: lv_lgpla.
137     MOVE ivhead-lgpla_verif TO lv_lgpla.
138
139     REFRESH: lt_bapiret.
140     CLEAR l_mtype.
141     CALL FUNCTION 'ZSPC_RF_SAVE_ZIVCOU'
142     EXPORTING

```

```
142     iv_lgpla      = lv_lgpla
143     is_item_head  = ls_item_head
144     iv_matnr      = ls_count-matnr
145     ivhead= ivhead
146     IMPORTING
147     et_bapiret    = lt_bapiret
148     e_rc_severity = l_mtype.
149
150     PERFORM error_analysis
151     USING l_mtype
152     lt_bapiret.
153
154     ...
```

ZSPC_RF_SAVE_ZIVCOU

```
1 FUNCTION zspc_rf_save_zivcou.
2 *"-----
3 *"*Local Interface:
4 *"  IMPORTING
5 *"    REFERENCE(IV_LGPLA) TYPE  /SCWM/LGPLA
6 *"    REFERENCE(IS_ITEM_HEAD) TYPE  /LIME/
7 *"    PI_ITEM_POS
8 *"    REFERENCE(IV_MATNR) TYPE  /SCWM/DE_MATNR
9 *"    REFERENCE(IVHEAD) TYPE  /SCWM/
10 *"    S_RF_INVENTORY_HEAD
11 *"  EXPORTING
12 *"    REFERENCE(ET_BAPIRET) TYPE  BAPIRET2_T
13 *"    REFERENCE(E_RC_SEVERITY) TYPE  BAPI_MTYPE
14 *"-----
15
16 DATA: lt_zspc_parm TYPE TABLE OF zspc_parm,
17        ls_zspc_parm LIKE LINE OF lt_zspc_parm,
18        lt_bapiret  TYPE  bapiret2_t,
19        l_mtype     TYPE  bapi_mtype.
20
21 DATA: ls_lagp      TYPE  /scwm/lagp,
22        lv_lgtyp    TYPE  /scwm/lgtyp,
23        lv_zparm_name TYPE  string,
24        ls_head_attr TYPE  /lime/
25        pi_head_attributes,
26        lt_item_to_read TYPE  /lime/pi_t_item_read
27        ,
28        ls_item_to_read  LIKE LINE OF
29        lt_item_to_read,
30        lt_item_to_delete TYPE  /lime/
31        pi_t_item_delete,
32        ls_item_to_delete LIKE LINE OF
33        lt_item_to_delete,
34        lt_item_read     TYPE  /lime/
35        pi_t_item_read_getsingle,
```



```

27     ls_item_read      TYPE /lime/
        pi_item_read_get_single ,
28     ls_book          TYPE /lime/
        pi_item_sub_read_get_sin ,
29     ls_cnt_res       TYPE /lime/
        pi_item_sub_read_get_sin ,
30     ls_diff          TYPE /lime/
        pi_item_sub_read_get_sin ,
31     ls_quan_temp     TYPE /lime/pi_quan ,
32     lv_book_quan     TYPE /scwm/de_quantity ,
33     lv_count_res     TYPE /scwm/de_quantity ,
34     lv_total_diff    TYPE /scwm/de_quantity ,
35     lv_threshold     TYPE /scwm/de_quantity ,
36     lv_over          TYPE xfeld VALUE
        abap_false .
37     CONSTANTS: lc_zparam_threshold TYPE string VALUE '
        INVE_THRESHOLD_'.
38     DATA: ls_post_head TYPE /lime/pi_head ,
39           lt_post_item  TYPE /lime/pi_t_item_post ,
40           ls_post_item  LIKE LINE OF lt_post_item ,
41           ls_timestamp_r TYPE timestampl ,
42           lt_pi_doc     TYPE /lime/pi_t_item_read ,
43           l_rc_severity TYPE bapi_mtype .
44     DATA: lt_zspc_phy_doc TYPE TABLE OF zspc_phy_doc ,
45           ls_zspc_phy_doc LIKE LINE OF
        lt_zspc_phy_doc ,
46           lv_pi_aread   TYPE /lime/pi_de_pi_aread ,
47           s_logitem     TYPE /lime/pi_doc_logitem .
48     DATA: ls_header    TYPE /scwm/bapi_pi_head ,
49           lt_keys       TYPE /scwm/t_bapi_pi_key ,
50           ls_keys       LIKE LINE OF lt_keys ,
51           lt_pi_count_item TYPE /scwm/
        t_bapi_pi_count_item ,
52           ls_pi_count_item LIKE LINE OF
        lt_pi_count_item ,
53           lt_return     TYPE bapirettab ,
54           lr_lgpla      TYPE rseloption ,

```

```

55         lrs_lgpla          LIKE LINE OF lr_lgpla,
56         lt_huitm          TYPE /scwm/tt_stock_select
57         ,
58         ls_huitm          LIKE LINE OF lt_huitm,
59         lv_matid_temp     TYPE /sapapo/matid.
60     " Recupero i dati della bin che sto contando
61     CALL FUNCTION '/SCWM/LAGP_READ_SINGLE'
62     EXPORTING
63         iv_lgnum          = gv_lgnum
64         iv_lgpla          = iv_lgpla
65     IMPORTING
66         es_lagp           = ls_lagp
67     EXCEPTIONS
68         wrong_input      = 1
69         not_found        = 2
70         enqueue_error    = 3
71         OTHERS           = 4.
72     IF sy-subrc <> 0.
73 *Implement suitable error handling here
74     ENDIF.
75
76     " Recupero lo storage type
77     CLEAR lv_lgtyp.
78     MOVE ls_lagp-lgtyp TO lv_lgtyp.
79
80     " Costruisco il nome del parametro soglia
81     CLEAR lv_zparm_name.
82     CONCATENATE lc_zparm_threshold lv_lgtyp INTO
83         lv_zparm_name.
84
85     " Recupero la riga contenente la soglia che mi
86     interessa
87     SELECT SINGLE *
88     FROM zspc_parm
89     INTO ls_zspc_parm
90     WHERE

```

```

89         zproc = gv_lgnum AND
90         repid = 'ZSPC_INVE_AUTO_POST' AND
91         zparm = lv_zparm_name.
92
93     IF sy-subrc <> 0.
94         "Errore in lettura
95     ENDIF.
96
97     " Recupero il documento di inventario completo
98     CLEAR ls_head_attr.
99     ls_head_attr-lgnum = is_item_head-lgnum.
100    ls_head_attr-process_type = is_item_head-
        process_type.
101
102    REFRESH lt_item_to_read.
103    CLEAR ls_item_to_read.
104    ls_item_to_read-doc_number = is_item_head-
        doc_number.
105    ls_item_to_read-doc_year = is_item_head-doc_year.
106    ls_item_to_read-item_no = is_item_head-item_no.
107    APPEND ls_item_to_read TO lt_item_to_read.
108
109    REFRESH: lt_bapiret, lt_item_read.
110    CLEAR l_rc_severity.
111    CALL FUNCTION '/SCWM/PI_CALL_DOCUMENT_READ_SI'
112        EXPORTING
113            is_head          = ls_head_attr
114            it_item          = lt_item_to_read
115        IMPORTING
116            et_item_read    = lt_item_read
117            et_bapiret      = lt_bapiret
118            e_rc_severity   = l_rc_severity.
119
120    CLEAR ls_item_read.
121    READ TABLE lt_item_read INTO ls_item_read INDEX
        1.
122

```

```

123     " Controllo sui materiali scansionati nella bin,
        devono essere uguali per le conte delle HU
124
125     " Recupero le conte per questo pi item
126     CLEAR: ls_header , ls_keys .
127     REFRESH lt_keys .
128     ls_header-whsenum = ls_item_read-data-lgnum .
129     MOVE-CORRESPONDING ls_item_read-data TO ls_keys .
130     APPEND ls_keys TO lt_keys .
131
132     REFRESH: lt_pi_count_item , lt_return .
133     CALL FUNCTION '/SCWM/BAPI_PI_READ_FOR_COUNT'
134         EXPORTING
135             is_header          = ls_header
136             it_keys            = lt_keys
137         IMPORTING
138             et_pi_count_item = lt_pi_count_item
139             et_return         = lt_return .
140
141     REFRESH lr_lgpla .
142     CLEAR lrs_lgpla .
143     lrs_lgpla-sign = wmegc_sign_inclusive .
144     lrs_lgpla-option = wmegc_option_eq .
145     lrs_lgpla-low = iv_lgpla .
146     lrs_lgpla-high = iv_lgpla .
147     APPEND lrs_lgpla TO lr_lgpla .
148
149     CALL FUNCTION '/SCWM/SELECT_STOCK'
150         EXPORTING
151             iv_lgnum = ls_item_read-data-lgnum
152             ir_lgpla = lr_lgpla
153         IMPORTING
154             et_huitm = lt_huitm
155         EXCEPTIONS
156             error    = 1
157             OTHERS  = 2 .
158     IF sy-subrc <> 0 .

```

```

159     ENDIF.
160
161     CLEAR: ls_pi_count_item.
162     LOOP AT lt_pi_count_item INTO ls_pi_count_item
163         WHERE item_material IS NOT INITIAL.
164         CALL FUNCTION 'CONVERSION_EXIT_MATID_INPUT'
165             EXPORTING
166                 input = ls_pi_count_item-item_material
167             IMPORTING
168                 output = lv_matid_temp.
169         READ TABLE lt_huitm INTO ls_huitm WITH KEY
170             matid = lv_matid_temp.
171
172         IF sy-subrc <> 0.
173             lv_over = abap_true.
174             EXIT.
175         ENDIF.
176     ENDL LOOP.
177
178     " Recupero la quantita' in stock
179     CLEAR:lv_book_quan, ls_book, ls_quan_temp.
180     LOOP AT ls_item_read-t_book INTO ls_book WHERE
181         t_quan IS NOT INITIAL.
182         READ TABLE ls_book-t_quan INTO ls_quan_temp
183             INDEX 1.
184         lv_book_quan = lv_book_quan + ls_quan_temp-
185             quantity.
186     ENDL LOOP.
187
188     " Recupero la quantita' di stock contata per ogni
189     hu e la sommo
190     CLEAR:lv_count_res, ls_cnt_res, ls_quan_temp.
191     LOOP AT ls_item_read-t_count_result INTO
192         ls_cnt_res WHERE t_quan IS NOT INITIAL.
193         READ TABLE ls_cnt_res-t_quan INTO ls_quan_temp
194             INDEX 1.

```

```

187     lv_count_res = lv_count_res + ls_quan_temp-
        quantity.
188 ENDLOOP.
189
190 " Calcolo la soglia in quantita' a partire dalla
        percentuale
191 lv_threshold = lv_book_quan / 100 * ls_zspc_parm-
        zlow.
192
193 " Calcolo la differenza totale, contando i segni
194 CLEAR: lv_total_diff, ls_diff, ls_quan_temp.
195 LOOP AT ls_item_read-t_difference INTO ls_diff.
196     READ TABLE ls_diff-t_quan INTO ls_quan_temp
        INDEX 1.
197     IF ls_diff-data-dif_direction EQ '0'.
198         lv_total_diff = lv_total_diff - ls_quan_temp-
            quantity.
199     ELSE.
200         lv_total_diff = lv_total_diff + ls_quan_temp-
            quantity.
201     ENDIF.
202 ENDLOOP.
203
204 IF lv_total_diff GE lv_threshold.
205     lv_over = abap_true.
206 ENDIF.
207
208 IF lv_over EQ abap_true.
209     " Canello il documento di inventario
210     REFRESH lt_item_to_delete.
211     CLEAR ls_item_to_delete.
212     ls_item_to_delete-doc_number = ls_item_read-
        data-doc_number.
213     ls_item_to_delete-doc_year = ls_item_read-data-
        doc_year.
214     ls_item_to_delete-item_no = ls_item_read-data-
        item_no.

```

```

215     APPEND ls_item_to_delete TO lt_item_to_delete.
216
217     REFRESH: et_bapiret, lt_item_to_read.
218     CLEAR e_rc_severity.
219     CALL FUNCTION '/SCWM/PI_CALL_DOCUMENT_DELETE'
220         EXPORTING
221             is_head      = ls_head_attr
222             it_item      = lt_item_to_delete
223         IMPORTING
224             et_pi_doc    = lt_item_to_read
225             et_bapiret   = et_bapiret
226             e_rc_severity = e_rc_severity.
227
228     IF NOT l_rc_severity CA wmegc_severity_ea.
229         COMMIT WORK AND WAIT.
230
231     REFRESH: et_bapiret, lt_item_read.
232     CLEAR e_rc_severity.
233     CALL FUNCTION '/SCWM/PI_CALL_DOCUMENT_READ_SI
234         ,
235         EXPORTING
236             is_head      = ls_head_attr
237             it_item      = lt_item_to_read
238         IMPORTING
239             et_item_read = lt_item_read
240             et_bapiret   = et_bapiret
241             e_rc_severity = e_rc_severity.
242
243     CLEAR ls_item_read.
244     READ TABLE lt_item_read INTO ls_item_read
245         INDEX 1.
246
247     "Recupero physical inventory area PI_AREAD
248     CALL METHOD grefo_cust->get_pi_aread
249         EXPORTING
250             i_pi_area = ls_item_read-data-pi_area
251         IMPORTING

```

```

250         e_pi_aread = lv_pi_aread.
251
252     "Aggiunta riferimenti alla tabella custom
      ZSPC_PHY_DOC
253     CLEAR ls_zspc_phy_doc.
254     DATA(item_data) = ls_item_read-data.
255     MOVE-CORRESPONDING item_data TO
      ls_zspc_phy_doc.
256     ls_zspc_phy_doc-pi_aread = lv_pi_aread.
257     ls_zspc_phy_doc-book_quantity = lv_book_quan.
258     ls_zspc_phy_doc-counted_quantity =
      lv_count_res.
259     ls_zspc_phy_doc-diff_quantity = lv_total_diff
      .
260     ls_zspc_phy_doc-threshold_qty = lv_threshold.
261
262     " Devo controllare se ci sono riferimenti
      inseriti nel documento
263     CLEAR s_logitem.
264     LOOP AT ls_item_read-t_logitem INTO s_logitem
      .
265         IF s_logitem-ref_doc_type EQ 'SCWM-RFUI'.
266             SPLIT s_logitem-ref_doc_id AT '-'
267                 INTO ls_zspc_phy_doc-doc_year_ref
      ls_zspc_phy_doc-doc_ref
      ls_zspc_phy_doc-item_ref.
268         ENDIF.
269     ENDLOOP.
270
271     ls_zspc_phy_doc-product = iv_matnr.
272     ls_zspc_phy_doc-lgpla = iv_lgpla.
273     CONCATENATE ls_item_read-data-lgnum ivhead-
      whord INTO ls_zspc_phy_doc-who.
274
275     INSERT zspc_phy_doc FROM ls_zspc_phy_doc.
276     ENDIF.
277 ELSE.

```



```

278     "Posting automatico
279     CLEAR ls_post_head.
280     ls_post_head-lgnum = is_item_head-lgnum.
281     ls_post_head-process_type = is_item_head-
        process_type.
282
283     REFRESH lt_post_item.
284     CLEAR ls_post_item.
285     ls_post_item-doc_number = is_item_head-
        doc_number.
286     ls_post_item-doc_year = is_item_head-doc_year.
287     ls_post_item-item_no = is_item_head-item_no.
288     CLEAR ls_timestamp_r.
289     GET TIME STAMP FIELD ls_timestamp_r.
290     ls_post_item-post_date = ls_timestamp_r.
291     APPEND ls_post_item TO lt_post_item.
292
293     REFRESH et_bapiret.
294     CALL FUNCTION '/SCWM/PI_CALL_DOCUMENT_POST'
295         EXPORTING
296             is_head      = ls_post_head
297             it_item      = lt_post_item
298         IMPORTING
299             et_pi_doc    = lt_pi_doc
300             et_bapiret  = et_bapiret
301             e_rc_severity = e_rc_severity.
302
303     IF NOT l_rc_severity CA wmegc_severity_ea.
304         COMMIT WORK AND WAIT.
305     ENDIF.
306 ENDIF.
307
308 ENDFUNCTION.

```

ZSPC_MANAGE_INVE_DOC_M

```
1  *&-----*
2  *& Report  ZSPC_MANAGE_INVE_DOC_M
3  *&
4  *&-----*
5  *&
6  *&
7  *&-----*
8  REPORT zspc_manage_inve_doc_m.
9
10 DATA: gr_alv          TYPE REF TO cl_salv_table ,
11         gr_functions  TYPE REF TO
12         cl_salv_functions_list ,
13         gr_columns    TYPE REF TO
14         cl_salv_columns_table ,
15         gr_column     TYPE REF TO cl_salv_column ,
16         gr_select     TYPE REF TO cl_salv_selections ,
17         gr_display    TYPE REF TO
18         cl_salv_display_settings ,
19         gr_layout     TYPE REF TO cl_salv_layout ,
20         lv_title      TYPE lvc_title ,
21         ls_column_ref TYPE salv_s_column_ref .
22 DATA: gt_zspc_phy_doc TYPE TABLE OF zspc_phy_doc .
23 DATA: gr_container  TYPE REF TO
24         cl_gui_custom_container .
25
26 "Costanti
27 CONSTANTS: gc_status_first  TYPE string VALUE '
28             FIRST' ,
29             gc_status_second TYPE string VALUE '
30             SECOND' ,
31             gc_status_third  TYPE string VALUE '
32             THIRD' .
33
34 * Selection screen
```

```

28 PARAMETERS: gv_lgnum TYPE zspc_phy_doc -lgnum
      VISIBLE LENGTH 4.
29 DATA: gv_doc_n TYPE zspc_phy_doc -doc_number ,
30       gv_item  TYPE zspc_phy_doc -item_no ,
31       gv_doc_y TYPE zspc_phy_doc -doc_year ,
32       gv_doc_s TYPE zspc_phy_doc -doc_status ,
33       gv_bin   TYPE zspc_phy_doc -lgpla ,
34       gv_mat   TYPE zspc_phy_doc -product ,
35       gv_c_dat TYPE zspc_phy_doc -create_date .
36 SELECT -OPTIONS:
37     gr_doc_n FOR gv_doc_n ,
38     gr_item  FOR gv_item ,
39     gr_doc_y FOR gv_doc_y ,
40     gr_doc_s FOR gv_doc_s ,
41     gr_bin   FOR gv_bin ,
42     gr_mat   FOR gv_mat ,
43     gr_c_dat FOR gv_c_dat .
44
45 " Event handler class
46 CLASS gcl_event_handler DEFINITION .
47     PUBLIC SECTION .
48     METHODS :
49         on_user_command FOR EVENT added_function OF
      cl_salv_events
50         IMPORTING e_salv_function .
51 ENDCLASS .
52
53 CLASS gcl_event_handler IMPLEMENTATION .
54     METHOD on_user_command .
55         PERFORM create_new_document USING
      e_salv_function .
56     ENDMETHOD .
57 ENDCLASS .
58
59 DATA: gr_events TYPE REF TO gcl_event_handler .
60
61 CLASS demo DEFINITION .

```

```

62     PUBLIC SECTION.
63         CLASS-METHODS main.
64     ENDCLASS.
65
66     CLASS demo IMPLEMENTATION.
67         METHOD main.
68
69             DATA: ls_zspc_phy_doc    LIKE LINE OF
70                   gt_zspc_phy_doc ,
71                   lt_stock_for_area TYPE /scwm/
72                   t_pi_stock_for_area ,
73                   lv_to_delete      TYPE xfeld.
74             DATA: lv_datlo          LIKE sy-datlo ,
75                   lv_timlo         LIKE sy-timlo ,
76                   lv_create_date   TYPE p ,
77                   lv_count_date    TYPE p.
78
79             SELECT *
80                 FROM zspc_phy_doc
81                 INTO TABLE gt_zspc_phy_doc
82                 WHERE doc_number IN gr_doc_n
83                   AND item_no IN gr_item
84                   AND doc_year IN gr_doc_y
85                   AND doc_status IN gr_doc_s
86                   AND lgpla IN gr_bin
87                   AND product IN gr_mat
88                   AND create_date IN gr_c_dat.
89
90             "Controllo lo stato dei documenti e cancello
91             quelli ACTI-POST
92             CLEAR ls_zspc_phy_doc.
93             LOOP AT gt_zspc_phy_doc INTO ls_zspc_phy_doc.
94                 CLEAR lv_to_delete.
95                 PERFORM check_doc_status
96                     USING
97                         ls_zspc_phy_doc
98                     CHANGING

```

```

96         lv_to_delete.
97     IF lv_to_delete = abap_true.
98         DELETE FROM zspc_phy_doc
99         WHERE lgnum = ls_zspc_phy_doc - lgnum
100            AND doc_number = ls_zspc_phy_doc -
101                doc_number
102            AND doc_year = ls_zspc_phy_doc - doc_year
103            AND item_no = ls_zspc_phy_doc - item_no.
104     ENDIF.
105     ENDLOOP.
106     "Recupero nuovamente le entry dalla tabella ora
107     "che ho cancellato le
108     "ACTI-POST
109     REFRESH gt_zspc_phy_doc.
110     SELECT *
111     FROM zspc_phy_doc
112     INTO TABLE gt_zspc_phy_doc
113     WHERE doc_number IN gr_doc_n
114            AND item_no IN gr_item
115            AND doc_year IN gr_doc_y
116            AND doc_status IN gr_doc_s
117            AND lgpla IN gr_bin
118            AND product IN gr_mat
119            AND create_date IN gr_c_dat.
120     "Controllo per ogni documento se il materiale
121     "sta venendo contato in
122     "un altro documento
123     CLEAR: ls_zspc_phy_doc.
124     LOOP AT gt_zspc_phy_doc INTO ls_zspc_phy_doc.
125         " Setto lo stato del documento FIRST, SECOND
126         " o THIRD
127         IF ls_zspc_phy_doc - doc_ref IS INITIAL.
128             ls_zspc_phy_doc - status_cr_count =
129                 gc_status_first.
130         ELSE.

```

```

128         IF ls_zspc_phy_doc-doc_type = 'HL'.
129             ls_zspc_phy_doc-status_cr_count =
                gc_status_second.
130         ELSE.
131             ls_zspc_phy_doc-status_cr_count =
                gc_status_third.
132         ENDIF.
133     ENDIF.
134
135     PERFORM check_already_counting
136         CHANGING
137             ls_zspc_phy_doc
138             lt_stock_for_area.
139
140     "Conversione dei timestamp
141     lv_count_date = ls_zspc_phy_doc-count_date.
142     lv_create_date = ls_zspc_phy_doc-create_date.
143
144     CLEAR: lv_datlo, lv_timlo.
145     CALL FUNCTION 'IB_CONVERT_FROM_TIMESTAMP'
146         EXPORTING
147             i_timestamp = lv_count_date
148             i_tzone      = sy-zonlo
149         IMPORTING
150             e_datlo      = lv_datlo
151             e_timlo      = lv_timlo.
152     ls_zspc_phy_doc-count_date_s = lv_datlo.
153     ls_zspc_phy_doc-count_time_s = lv_timlo.
154
155     CLEAR: lv_datlo, lv_timlo.
156     CALL FUNCTION 'IB_CONVERT_FROM_TIMESTAMP'
157         EXPORTING
158             i_timestamp = lv_create_date
159             i_tzone      = sy-zonlo
160         IMPORTING
161             e_datlo      = lv_datlo
162             e_timlo      = lv_timlo.

```

```
163         ls_zspc_phy_doc-create_date_s = lv_datlo.
164         ls_zspc_phy_doc-create_time_s = lv_timlo.
165
166         MODIFY gt_zspc_phy_doc FROM ls_zspc_phy_doc.
167     ENDLOOP.
168
169     CALL SCREEN 9999.
170 ENDMETHOD.
171 ENDCLASS.
172
173 START-OF-SELECTION.
174     demo⇒main( ).
175
176 INCLUDE zspc_manage_inve_doc_m_f01.
```

ZSPC_MANAGE_INVE_DOC_M_F01

```
1 *-----*
2 ***INCLUDE ZSPC_MANAGE_INVE_DOC_M_F01.
3 *-----*
4 *&-----*
5 *&      Form  CHECK_DOC_STATUS
6 *&-----*
7 *      text
8 *-----*
9 *  →  p1      text
10 *  <-- p2     text
11 *-----*
12 FORM check_doc_status
13 USING is_zspc_phy_doc TYPE zspc_phy_doc
14 CHANGING cv_to_delete TYPE xfeld.
15
16 DATA: ls_head_attr    TYPE /lime/
17        pi_head_attributes ,
18        lt_item_to_read TYPE /lime/pi_t_item_read ,
19        ls_item_to_read LIKE LINE OF
20        lt_item_to_read ,
21        lt_item_read    TYPE /lime/
22        pi_t_item_read_getsingle ,
23        ls_item_read    TYPE /lime/
24        pi_item_read_get_single ,
25        lt_bapiret      TYPE bapiret2_t ,
26        l_mtype         TYPE bapi_mtype.
27
28 " Recupero il documento di inventario completo
29 CLEAR ls_head_attr.
30 ls_head_attr-lgnum = is_zspc_phy_doc-lgnum.
31 ls_head_attr-process_type = is_zspc_phy_doc-
32 process_type.
33
34 REFRESH lt_item_to_read.
35 CLEAR ls_item_to_read.
```



```

31   ls_item_to_read-doc_number = is_zspc_phy_doc -
      doc_number.
32   ls_item_to_read-doc_year = is_zspc_phy_doc -
      doc_year.
33   ls_item_to_read-item_no = is_zspc_phy_doc-item_no
      .
34   APPEND ls_item_to_read TO lt_item_to_read.
35
36   REFRESH: lt_bapiret, lt_item_read.
37   CLEAR l_mtype.
38   CALL FUNCTION '/LIME/PI_DOCUMENT_READ_SINGLE'
39     EXPORTING
40       is_head      = ls_head_attr
41       it_item      = lt_item_to_read
42     IMPORTING
43       et_item_read = lt_item_read
44       et_bapiret  = lt_bapiret
45       e_rc_severity = l_mtype.
46
47   READ TABLE lt_item_read INTO ls_item_read INDEX
      1.
48
49   IF ls_item_read-data-doc_status EQ 'ACTI' OR
      ls_item_read-data-doc_status EQ 'POST'.
50     cv_to_delete = abap_true.
51   ELSEIF ls_item_read-data-doc_type EQ 'HS' AND
      ls_item_read-data-doc_status EQ 'DELE'.
52     cv_to_delete = abap_true.
53   ELSE.
54     cv_to_delete = abap_false.
55   ENDIF.
56
57   ENDFORM.
58   *&-----*
59   *&      Module s9999_pbo OUTPUT
60   *&-----*
61   *      text

```

```

62 *-----*
63 MODULE s9999_pbo OUTPUT.
64
65     SET PF-STATUS 'S9999'.
66
67     " Creazione dell'oggetto che contiene il
        riferimento al container
68     IF gr_container IS NOT BOUND.
69         CREATE OBJECT gr_container
70             EXPORTING
71                 container_name = 'ALV_CONTAINER'.
72     ENDIF.
73
74     " Creazione dell'ALV
75     TRY.
76         cl_salv_table=>factory(
77             EXPORTING
78                 r_container = gr_container
79                 container_name = 'ALV_CONTAINER'
80             IMPORTING r_salv_table = gr_alv
81             CHANGING t_table = gt_zspc_phy_doc ).
82     CATCH cx_salv_msg .
83     ENDTRY.
84     IF gr_alv IS INITIAL.
85         MESSAGE 'Error Creating ALV Grid ' TYPE 'I'
            DISPLAY LIKE 'E'.
86     ENDIF.
87     " Rendo visibili tutte le funzioni dell'ALV
88     gr_functions = gr_alv->get_functions( ).
89     gr_functions->set_all( if_salv_c_bool_sap=>true ).
90     TRY .
91         gr_functions->add_function(
92             EXPORTING
93                 name = 'FUNC_CREA_DOC'
94                 text = 'Create Inventory Document'
95                 tooltip = 'Create Inventory Document'
96                 position =

```

```

97     if_salv_c_function_position⇒
          right_of_salv_functions ).
98     CATCH cx_salv_existing cx_salv_wrong_call.
99     ENTRY.
100    " Global display settings
101    CLEAR gr_display.
102    MOVE 'SPC: Physical inventory monitor' TO
          lv_title.
103    gr_display = gr_alv→get_display_settings( ).
104    gr_display→set_list_header( lv_title ).
105    " Manage row selection
106    gr_select = gr_alv→get_selections( ).
107    gr_select→set_selection_mode(
          if_salv_c_selection_mode⇒row_column ).
108    "Allow single row Selection"
109    " Optimize column width
110    CLEAR gr_columns.
111    gr_columns = gr_alv→get_columns( ).
112    gr_columns→set_optimize( if_salv_c_bool_sap⇒true
          ).
113    " Nascondere le colonne che non devono essere
          visibili
114    CLEAR gr_column.
115    gr_column = gr_columns→get_column( 'CREATE_DATE'
          ).
116    gr_column→set_visible( if_salv_c_bool_sap⇒false )
          .
117    CLEAR gr_column.
118    gr_column = gr_columns→get_column( 'COUNT_DATE' )
          .
119    gr_column→set_visible( if_salv_c_bool_sap⇒false )
          .
120    " Register events to event handler
121    CREATE OBJECT gr_events.
122    SET HANDLER gr_events→on_user_command FOR gr_alv→
          get_event( ).
123    " Render dell'ALV

```

```

124     gr_alv→display( ).
125 ENDMODULE.
126
127 *&-----*
128 *&       Module  s9999_pai  INPUT
129 *&-----*
130 *       text
131 *-----*
132 MODULE s9999_pai INPUT.
133     CASE sy-ucomm.
134         WHEN 'BACK'.
135             LEAVE TO SCREEN 0.
136         WHEN 'EXIT'.
137             LEAVE PROGRAM.
138         WHEN 'CANCEL'.
139             LEAVE PROGRAM.
140     ENDCASE.
141 ENDMODULE.
142 *&-----*
143 *&       Form  create_new_document
144 *&-----*
145 *       text
146 *-----*
147 *   →  p1           text
148 *  <-- p2           text
149 *-----*
150 FORM create_new_document USING i_function TYPE
    salv_de_function.
151
152     DATA: lt_sel_rows      TYPE salv_t_row,
153           ls_sel_rows      LIKE LINE OF lt_sel_rows,
154           ls_zspc_phy_doc  LIKE LINE OF
    gt_zspc_phy_doc,
155           lt_stock_for_area TYPE /scwm/
    t_pi_stock_for_area.
156

```

```

157 DATA: ls_create_header      TYPE /scwm/
      bapi_pi_create_head ,
158      lt_create_item          TYPE TABLE OF /scwm/
      bapi_pi_create_item ,
159      ls_create_item          LIKE LINE OF
      lt_create_item ,
160      lt_create_reference     TYPE TABLE OF /scwm/
      bapi_pi_create_reference ,
161      ls_create_reference     LIKE LINE OF
      lt_create_reference ,
162      ls_stock_range          TYPE /scwm/
      s_stock_range ,
163      lr_matnr                TYPE rseloption ,
164      lrs_matnr               LIKE LINE OF lr_matnr ,
165      ls_loc_range            TYPE /scwm/
      s_pi_loc_range ,
166      lr_lgpla                TYPE rseloption ,
167      lrs_lgpla               LIKE LINE OF lr_lgpla ,
168      lt_keys                 TYPE /scwm/
      t_bapi_pi_key ,
169      lt_bapiret              TYPE bapiret2_t ,
170      l_rc_severity           TYPE mtype2 ,
171      lv_answer                TYPE string.
172
173 REFRESH lt_sel_rows.
174 lt_sel_rows = gr_select->get_selected_rows( ).
175
176 IF lt_sel_rows IS NOT INITIAL.
177     "Recupero la riga della tabella selezionata
178     READ TABLE lt_sel_rows INTO ls_sel_rows INDEX
179     1.
180     "Recupero il contenuto della riga selezionata
181     READ TABLE gt_zspc_phy_doc INTO ls_zspc_phy_doc
182     INDEX ls_sel_rows.

```

```

183     IF ls_zspc_phy_doc-status_icon EQ
184         icon_green_light.
185         "Popup di conferma della creazione del nuovo
186         documento
187         CALL FUNCTION 'POPUP_TO_CONFIRM'
188         EXPORTING
189             titlebar           = text-001
190             text_question      = text-002
191             text_button_1      = text-003
192             text_button_2      = text-004
193             default_button     = '2'
194             display_cancel_button = ''
195             popup_type         = '
196                 ICON_MESSAGE_WARNING'
197         IMPORTING
198             answer              = lv_answer.
199
200     "Se la risposta e' no ritorno
201     IF lv_answer EQ '2'.
202         RETURN.
203     ENDIF.
204
205     "Ricontrollo se al momento ci sono altri
206     documenti di conta per il materiale e allo
207     "stesso tempo recupero gia' le info che mi
208     servono sullo stock per creare i nuovi doc
209     PERFORM check_already_counting
210     CHANGING
211         ls_zspc_phy_doc
212         lt_stock_for_area.
213
214     "Se il semaforo e' ancora verde posso
215     procedere alla creazione del nuovo
216     documento
217     IF ls_zspc_phy_doc-status_icon EQ
218         icon_green_light.
219         "Gestisco i vari stati del documento

```

```

212     CASE ls_zspc_phy_doc-status_cr_count.
213         WHEN gc_status_first.
214             "Documento HL alla prima conta, creo un
                nuovo HL per quella bin
215
216             CLEAR ls_create_header.
217             ls_create_header-procedure = 'HL'.
218             ls_create_header-whsnumber =
                ls_zspc_phy_doc-lgnum.
219
220             CLEAR ls_create_item.
221             REFRESH lt_create_item.
222
223             CALL FUNCTION '/SCWM/
                BAPI_PI_DOCUMENT_CREATE'
224             EXPORTING
225                 create_header      = ls_create_header
226             IMPORTING
227                 et_keys            = lt_keys
228             TABLES
229                 create_item       = lt_create_item
230                 create_reference  =
                lt_create_reference
231             RETURN                  = lt_bapiret.
232
233         WHEN gc_status_second.
234             "Documento HL alla seconda conta, creao
                un nuovo HS per il prodotto
235     CLEAR ls_create_header.
236         ls_create_header-procedure = 'HS'.
237         ls_create_header-whsnumber =
                ls_zspc_phy_doc-lgnum.
238
239         CLEAR ls_create_item.
240         REFRESH lt_create_item.
241

```

```

242         CALL FUNCTION '/SCWM/
           BAPI_PI_DOCUMENT_CREATE'
243         EXPORTING
244             create_header      = ls_create_header
245         IMPORTING
246             et_keys            = lt_keys
247         TABLES
248             create_item        = lt_create_item
249             create_reference    =
250                 lt_create_reference
251             return              = lt_bapiret.
252     WHEN gc_status_third.
253         "Errore
254 RETURN .
255     ENDCASE.
256     ELSE.
257         "Errore, semaforo diventato rosso nel
258             frattempo
259         MESSAGE "Product already being counted"
260             TYPE 'E'.
261     ENDF.
262     ELSE.
263         "Altri documenti di inventario aperti per
264             questo prodotto, non accade nulla
265         MESSAGE "Product already being counted" TYPE
266             'E'.
267     ENDF.
268     ELSE.
269         MESSAGE 'No row selected' TYPE 'E'.
270     ENDF.
271 ENDFORM.
272 *&-----*
273 *&         Form   check_already_counting
274 *&-----*

```



```

273 *          text
274 *-----*
275 *   →  p1          text
276 *  <-- p2          text
277 *-----*
278 FORM check_already_counting
279   CHANGING  ls_zspc_phy_doc LIKE LINE OF
          gt_zspc_phy_doc
280           lt_stock_for_area TYPE /scwm/
          t_pi_stock_for_area.
281
282   DATA: ls_stock_range TYPE /scwm/s_stock_range ,
283         lr_matnr        TYPE rseloption ,
284         lrs_matnr       LIKE LINE OF lr_matnr ,
285         lt_bapiret      TYPE bapiret2_t ,
286         ls_bapiret      LIKE LINE OF lt_bapiret.
287
288   "Se non ho riferimenti ad altri documenti vuol
          dire che da una riga posso creare un HL,
289   "quindi controllo che non ci siano HL gia' creati
          per quella riga.
290   "Viceversa, se ci sono riferimenti il prossimo
          passo e' creare un HS quindi controllo che
291   "non ci sia gia' un HS creato per lo stesso
          materiale.
292
293   IF ls_zspc_phy_doc-doc_ref IS NOT INITIAL.
294     CLEAR: ls_stock_range , lrs_matnr.
295     REFRESH lr_matnr.
296
297     lrs_matnr-sign = wmegc_sign_inclusive.
298     lrs_matnr-option = wmegc_option_eq.
299     lrs_matnr-low = ls_zspc_phy_doc-product.
300     lrs_matnr-high = ls_zspc_phy_doc-product.
301     APPEND lrs_matnr TO lr_matnr.
302     ls_stock_range-r_matnr = lr_matnr.
303

```

```

304     REFRESH lt_bapiret.
305     CALL FUNCTION '/SCWM/PI_GET_STOCK_FOR_AREA'
306         EXPORTING
307             iv_lgnum           = ls_zspc_phy_doc -lgnum
308             iv_doc_type        = 'HS'
309             is_stock_range     = ls_stock_range
310         IMPORTING
311             et_stock_for_area = lt_stock_for_area
312             et_bapiret       = lt_bapiret.
313     ELSE.
314
315     ENDIF.
316
317     "Provo a recuperare l'elemento di tipo I nella
318     bapiret
319     READ TABLE lt_bapiret INTO ls_bapiret WITH KEY
320         type = 'I'.
321     IF sy-subrc = 0.
322         WRITE icon_red_light AS ICON TO ls_zspc_phy_doc
323             -status_icon.
324     ELSE.
325         WRITE icon_green_light AS ICON TO
326             ls_zspc_phy_doc -status_icon.
327     ENDIF.
328 ENDFORM.

```

Bibliografia

- [1] <http://www00.unibg.it/dati/corsi/87101/77954-Il%20magazzino.pdf>. [Online; accessed 2017]. 2017.
- [2] <http://simplestudies.com/when-is-physical-inventory-taken.html>. [Online; accessed 2017]. 2017.
- [3] <https://help.sap.com/>. [Online; accessed 2018]. 2018.
- [4] <https://www.tutorialspoint.com/>. [Online; accessed 2018]. 2018.
- [5] *art.1, D.P.R. 695/96 per la contabilità di magazzino, di cui all'art.14, c.1, lett. d.* 1996.
- [6] *EWM100 SAP course, svolto in sede.* 2018.
- [7] *EWM110 SAP course, svolto in sede.* 2018.
- [8] Julia Kirchner Gunther Farber. *ABAP Basics*. A cura di SAP PRESS. 2007.
- [9] Sascha Kruger Horst Keller. *ABAP Objects. ABAP programming in SAP NetWeaver*. A cura di SAPPROUK Limited. 2011.
- [10] Project Management Institute. *A Guide to the Project Management Body of Knowledge (Quinta edizione)*. A cura di Project Management Institute. 2013.
- [11] Peter Moxon. *Beginner's guide to SAP ABAP. An introduction to programming SAP applications using ABAP*. A cura di SAPPROUK Limited. 2012.