MINISTERO DELLO SVILUPPO ECONOMICO
DIREZIONE GENERALE PER LA LOTTA ALLA CONTRAFFAZIONE
UFFICIO ITALIANO BREVETTI E MARCHI

DOMANDA NUMERO	**102000900842508**
Data Deposito	**02/05/2000**
Data Pubblicazione	**02/11/2001**

Priorità	19922768.3
Nazione Priorità	DE
Data Deposito Priorità	

Sezione	**Classe**	**Sottoclasse**	**Gruppo**	**Sottogruppo**
G	06	F		

Titolo

PROCEDIMENTO PER L'INSTALLAZIONE DI SOFTWARE E/O PER LA PROVA DI UN SISTEMA DI ELABORATORE.

DESCRIZIONE dell'invenzione industriale dal titolo:
"Procedimento per l'installazione di software e/o per la prova di un sistema di elaboratore",
di: DELL USA, L.P., nazionalità statunitense, One Dell Way, Round Rock, Texas 78682-2244
Inventori designati: AMBERG, Richard, D.; WONG, Roger; LYNCH, Michael.
Depositata il: 2 MAGGIO 2000

TO 2000A 000409

* * *

TESTO DELLA DESCRIZIONE

SFONDO

Le descrizioni in questo contesto sono relative ad un procedimento per l'installazione di software e/o per la prova di un sistema di elaboratore.

La presente domanda è relativa alla Domanda di brevetto tedesco anch'essa pendente con il numero di serie 19836328.1, depositata l'11 agosto 1998, intitolata SOFTWARE INSTALLATION AND TESTING FOR A BUILD-TO-ORDER COMPUTER SYSTEM, che cita Richard D. Amberg, Roger W. Wong e Michael A. Brundridge come inventori.

La presente domanda è relativa alla Domanda di Brevetto tedesco anch'essa pendente con il numero di serie 19836333.8, depositata l'11 agosto 1998, intitolata SOFTWARE INSTALLATION AND TESTING FOR A BUILD-TO-ORDER COMPUTER SYSTEM, che cita Richard D.

Amberg, Roger W. Wong e Michael A. Brundridge come inventori.

La presente domanda è relativa alla Domanda di Brevetto tedesco anch'essa pendente con il numero di serie 19836381.8, depositata l'11 agosto 1998, intitolata DATABASE FOR FACILITATING SOFTWARE INSTALLATION AND TESTING FOR A BUILD-TO-ORDER COMPUTER SYSTEM, che cita Richard D. Amberg, Roger W. Wong e Michael A. Brundridge come inventori.

Queste domande pendenti sono qui allegate per riferimento nella loro interezza, e sono assegnate al titolare della presente invenzione.

I sistemi di elaboratori personali in generale e i sistemi di elaboratori personali IBM compatibili in particolare hanno ottenuto un utilizzo diffuso per fornire potenza di calcolo a molti segmenti della società. Un sistema di elaboratore personale può essere definito di solito come microelaboratore da scrivania, da pavimento o portatile che comprende una unità di sistema avente un processore di sistema ed una memoria volatile o non volatile associata, un monitor di visualizzazione, una tastiera, una o più unità a dischetti, un dispositivo di memorizzazione a disco fisso ed una stampante opzionale.

E' noto installare software ed eseguire test su sistemi di elaboratore prima che essi siano spediti

alle aziende o singoli clienti. L'obiettivo dell'installazione dei test software è quello di produrre in modo efficiente un sistema di elaboratore utile ed affidabile che possa essere inviato ad aziende ed individui privo di errori e pronto all'esecuzione. In generale, i test rilevano e analizzano errori che si verificano nelle porzioni sia hardware sia software del sistema di elaboratore. Un elenco parziale dei test hardware di sistema di elaboratore potrebbe comprendere diagnostica su componenti hardware quali processore, memoria, un dispositivo di memorizzazione su disco, un dispositivo audio, un dispositivo grafico, una tastiera, un mouse ed una stampante. L'installazione software comprende spesso il caricamento di un pacchetto desiderato di software sul sistema di elaboratore, la preparazione di variabili di ambiente appropriate per l'elaboratore, e la preparazione di file di inizializzazione appropriati per il software caricato. I test software comprendono spesso l'accertamento che è stata installata una versione desiderata di software sul sistema di elaboratore e che i dispositivi di comando appropriati sono presenti sul sistema di elaboratore.

E' noto nell'industria installare software e

testare sistemi di elaboratore durante la fabbricazione eseguendo una procedura fissa prima di spedirli ai clienti. Per esempio, si crea un dischetto contenente certi test diagnostici per un certo tipo di sistema di elaboratore. Il dischetto comprende file batch lunghi e spesso complicati che dirigono i processi di installazione e diagnostica di software. Il dischetto comprende inoltre tutti i file eseguibili per eseguire test sul sistema di elaboratore acquistato.

Ogni sistema di elaboratore acquistato è dotato di una rispettiva copia di questo dischetto. Questi dischetti accompagnano i sistemi di elaboratore costruiti lungo lo stabilimento durante il processo di fabbricazione, eseguendosi test sul rispettivo sistema di elaboratore secondo l'ordine inerente nel file batch. Se occorre effettuare una modifica al processo, il file batch viene modificato in modo corrispondente aggiungendo o togliendo porzioni dal codice batch. Tale variazione al file batch dà come risultato una corrispondente variazione ai parametri di test (compresa la sequenza in cui si eseguono i test) di ogni successivo sistema di elaboratore fabbricato, dato che ogni sistema di elaboratore condivide la stessa procedura diagnostica di file batch.

Mentre soluzioni diagnostiche di questo tipo hanno presentato un certo grado di utilità nell'aumentare l'affidabilità di sistemi di elaboratore prima della spedizione, rimane spazio per miglioramenti. Per esempio, dato che i test continuano a diventare sempre più complicati e approfonditi, i file batch e i file eseguibili dai test diagnostici superano spesso le capacità di memorizzazione di un dischetto. Inoltre, è spesso difficile o impossibile personalizzare le procedure di installazione software e di test per un singolo sistema di elaboratore costruito secondo un ordine o per una certa famiglia di sistemi di elaboratore senza modificare i test per altri sistemi e famiglie. Inoltre, è difficile o impossibile modificare l'ordine dell'installazione software o dei test per un singolo sistema di elaboratore costruito secondo un ordine o per una certa famiglia di sistemi di elaboratore senza modificare l'ordine per altri sistemi e famiglie. Infine, la natura spesso complicata delle strutture di file batch correnti rende talvolta difficile per i costruttori risolvere i problemi o mantenere le procedure di test di installazione software in maniera rapida ed efficiente.

Pertanto, quello che è necessario è prevedere

un procedimento per l'installazione di software e/o per la prova di un sistema di elaboratore che eviti i limiti associati alla tecnica anteriore.

L'invenzione

Una forma di attuazione di conseguenza prevede un procedimento per l'installazione di software su un sistema di elaboratore prevedendo una sequenza di fasi comprendenti una pluralità di fasi di installazione di software da eseguire in un ordine determinato dalla sequenza di fasi, e leggendo ed eseguendo fasi consecutive dalla sequenza di fasi.

Breve descrizione dei disegni

La Figura 1 è un diagramma schematico che illustra una forma di attuazione di un'installazione di software ed i test.

La Figura 2 è un diagramma schematico di una installazione di software e di test secondo un'altra forma di attuazione.

La Figura 3A è un diagramma di flusso che illustra una forma di attuazione per convertire un ordine di elaboratore in un record descrittore di sistema secondo la presente invenzione.

La Figura 3B illustra una forma di attuazione di una porzione di un ordine di elaboratore esemplificativo, un file di Record di Assemblaggio di Base (BAR), e un record descrittore di sistema.

La Figura 4 è un diagramma di flusso che illustra una forma di attuazione per creare e fornire una sequenza di fasi.

La Figura 5 è un diagramma di flusso più dettagliato che illustra una forma di attuazione per creare una sequenza di fasi.

La Figura 6 è un'illustrazione di una forma di attuazione di una struttura di una banca dati.

La Figura 7 è un esempio che illustra una forma di attuazione di parte di un file di fasi.

Le Figure da 8 a 13 sono diagrammi di flusso che illustrano una forma di attuazione del funzionamento di un programma per eseguire una sequenza di fasi.

Descrizione dettagliata

La Figura 1 è un diagramma schematico di un sistema di installazione software e di test 90 in un sito di fabbricazione di sistemi di elaboratore. Durante il funzionamento, si inserisce un ordine 92 per acquistare un sistema di elaboratore obiettivo costruito secondo un ordine 160. Il sistema obiettivo 160 deve essere fabbricato in modo tale da contenere una pluralità di componenti hardware e software. Per esempio, il sistema obiettivo 160 potrebbe comprendere una certa marca di unità a disco rigido, un particolare tipo di monitor, una

certa marca di processore, ed una particolare versione di un sistema operativo. Prima di spedire il sistema obiettivo 160 al cliente, si installano e si testano la pluralità di componenti. Tale installazione di software e test garantiscono in modo vantaggioso un sistema di elaboratore operativo ed affidabile che è pronto all'esecuzione quando viene ricevuto.

Poiché famiglie diverse di sistemi di elaboratore e componenti di elaboratore diversi richiedono fasi di installazione di software e di test diverse, è necessario determinare quali test occorre eseguire sul sistema obiettivo 160 e in quale ordine questi test devono essere eseguiti in modo da ottenere un processo di installazione di software e di test efficace. Il dispositivo realizzatore di fasi 140 è un sistema di elaboratore configurato in modo da sequenziare le fasi di installazione di software e di test da eseguire sul sistema obiettivo 160. Per sequenziare le fasi di installazione e di software e/o di test, il dispositivo di realizzazione di fasi 140, e più in particolare il programma di sequenziamento 204 che risiede sul dispositivo di realizzazione di fasi 140, legge dapprima una pluralità di descrittori di componenti dal file descrittore 96. Il file

descrittore 96 è previsto per convertire un ordine 92, che corrisponde ad un sistema di elaboratore desiderato avente componenti desiderati, in un formato leggibile da elaboratore tramite il modulo di conversione 94.

I descrittori di componenti sono descrizioni leggibili da elaboratore dei componenti del sistema obiettivo 160 i quali componenti sono definiti dall'ordine 92. Nella forma di attuazione preferita, i descrittori di componenti sono compresi in un file descrittore chiamato record descrittore di sistema che è un file leggibile da elaboratore contenente un listato dei componenti, dei componenti hardware e/o software, da installare sul sistema obiettivo 160. Avendo letto la pluralità di descrittori di componenti, il programma di sequenziamento 204 reperisce una pluralità di fasi di installazioni di software e/o di test corrispondenti ai descrittori di componenti dalla banca dati 100 sulla connessione di rete 110. La connessione di rete 110 può essere qualsiasi connessione di rete ben nota nella tecnica, quale una rete di area locale, una intranet o l'internet. Le informazioni contenute nella banca dati 100 possono essere aggiornate attraverso una modifica illustrata dalla freccia 130.

Avendo reperito le fasi di installazione

software e/o di test appropriate per il sistema obiettivo 160, il programma di sequenziamento 204 sequenzia le fasi in un predeterminato ordine secondo i numeri di sequenza corrispondenti ad ogni fase. Avendo sequenziato le fasi richieste per il sistema obiettivo 160, il programma di sequenziamento 204 scrive una serie di file di uscita sul disco di fasi 150. Nella forma di attuazione evidenziata in Figura 1, i file di uscita comprendono file di testo contenenti linee di comando appropriate per l'esecuzione delle fasi di installazione software e/o di test appropriate sul sistema obiettivo 160. L'esecuzione è effettuata nel predeterminato ordine secondo i numeri di sequenza corrispondenti ad ogni fase. Il disco di fase 150 accompagna il sistema obiettivo 160 in fabbrica dove i test vengono eseguiti direttamente dal disco di fase 150 o, in alternativa, dal server di file 190, connesso al sistema obiettivo 160 tramite la connessone di rete 180. Preferibilmente, la connessione di rete 180 è un dispositivo di rete generico collegato ad una corrispondente porta di rete del sistema elaboratore obiettivo. A seguito dell'esecuzione delle fasi di installazione software e di test, i risultati delle installazioni dei test sono registrati di nuovo sul server di file 190 e

sulla connessione di rete 180.

Preferibilmente, la connessione di rete 180 è un dispositivo di rete generico collegato ad una corrispondente porta di rete del sistema elaboratore obiettivo. A seguito dell'esecuzione delle fasi di installazione software e di test, i risultati delle installazioni dei test sono registrati di nuovo sul server di file 190 sulla connessione di rete 180.

La Figura 2 è un diagramma schematico del sistema di installazione software di test 192 in conformità con un'altra forma di attuazione della presente invenzione. Un cliente invia un ordine 92 per acquistare un sistema di elaboratore obiettivo 160 costruito secondo un ordine. Il sistema obiettivo 160 deve essere fabbricato in modo da contenere una pluralità di componenti i quali componenti possono comprendere componenti sia hardware sia/o software. Prima di spedire al cliente il sistema obiettivo 160, si installano e si testano la pluralità di componenti. Tale installazione e test garantisce in modo vantaggioso un sistema di elaboratore operativo ed affidabile che è pronto all'esecuzione quando è ricevuto dal cliente.

Per sequenziare le fasi di installazione di software e di test, il programma di sequenziamento 204 legge una pluralità di descrittori di componenti

dal file descrittore 96. L'ordine 92 viene convertito nel file descrittore 96 tramite il modulo di conversione 94. I descrittori di componenti sono descrizioni leggibili da elaboratore dei componenti del sistema obiettivo 160. Nella forma di attuazione preferita, i descrittori di componenti sono compresi in un file descrittore chiamato record descrittore di sistema, un file leggibile da elaboratore contenente un listato di ogni componente, sia hardware sia software, da installare sul sistema obiettivo 160. Il record descrittore di sistema può essere memorizzato direttamente sul server di file 202. Il programma di sequenziamento 204 reperisce una pluralità di fasi di installazione software e/o di test corrispondente ai descrittori di componenti dalla banca dati 100. Avendo reperito le fasi di installazione di software e/o di test appropriate per il sistema obiettivo 160, il programma di sequenziamento 204 sequenzia le fasi in un predeterminato ordine secondo i numeri di sequenziamento corrispondenti ad ogni fase. Avendo sequenziato le fasi richieste per il sistema obiettivo 160, il programma di sequenziamento 204 dirige l'esecuzione delle fasi di installazione di software e di test sul sistema obiettivo 160 nel predeterminato ordine attraverso le connessioni di

rete 195 e 180. Si desidera che la connessione di rete 200 sia un dispositivo di rete generico collegato in una corrispondente porta del sistema obiettivo 160. La rete 195 può essere qualsiasi connessione di comunicazione ben nota nella tecnica. A seguito dell'esecuzione delle fasi di installazione di software e/o di test, i risultati delle installazioni e dei test vengono registrati di nuovo nel server di fase 202 sulla connessione di rete 200 o memorizzati all'interno di una banca dati appropriata. Come appare evidente dall'illustrazione, non si ha alcuna necessità di un sistema di elaboratore 140 al dispositivo di realizzazione di fase separato di Figura 1. In aggiunta, il disco di fase 150 non è necessario. Piuttosto è necessario soltanto il disco di inizializzazione 220, che è configurato per inizializzare il sistema obiettivo 160, in modo che accompagni il sistema obiettivo 160 in fabbrica.

Avendo descritto in generale i sistemi di installazione di software e di test, si presterà ora attenzione a descrivere il funzionamento dei sistemi evidenziati delle Figure 1 e 2 in maggior dettaglio.

La Figura 3A illustra il processo preferito in cui ordine per un sistema di elaboratore viene convertito in un record descrittore di sistema

leggibile da elaboratore. Più specificamente, alla voce 300, si riceve un ordine per un sistema elaboratore obiettivo. Quest'ordine può essere in una qualsiasi di innumerevoli forme. Per esempio, sono possibili formati di ordinamento diversi così come i meccanismi di invio di ordine diversi. Per esempio, ordini per un sistema di elaboratore obiettivo possono essere inviati per telefono, tramite posta o su reti di elaboratore (ad esempio sull'internet). Indipendentemente dai mezzi di ricevere o dalla forma dell'ordine, l'ordine comprende il tipo di sistema elaboratore obiettivo che un cliente desidera acquistare e, eventualmente, un listato esplicito dei particolari componenti che il cliente desidera che siano compresi nel sistema elaboratore obiettivo. Dopo che si riceve l'ordine, il controllo effettua una transizione al modulo di trasmissione 310 durante il quale l'ordine di sistema di elaboratore obiettivo è trasmesso su una rete di elaboratori ad un sistema di fabbricazione (non illustrato) che produce il sistema di elaboratore obiettivo. L'ordine di sistema di elaboratore obiettivo è fornito inoltre al sistema di installazione software e di test dove viene incanalato in un programma di conversione nel modulo 320. La rete di elaboratori utilizzata nel modulo

310 può essere di qualsiasi tipo ben noto nella tecnica.

Il programma di conversione converte l'organo di sistema di elaboratore obiettivo in un record utile per il processo di fabbricazione. Più specificamente, il programma di conversione converte l'organo di elaboratore in primo luogo in un record chiamato un file BAR nel modulo 330. Preferibilmente il file BAR contiene un identificatore unico che identifica il sistema di elaboratore obiettivo specifico fabbricato. Il file BAR contiene inoltre un listato dettagliato dei componenti, che possono comprendere sia hardware, sia software da inserire nel sistema obiettivo. Inoltre, si desidera che il file BAR contenga numeri di particolari specifici del costruttore o altri identificatori utili per ogni componente. Infine, il file BAR può contenere informazioni specifiche del cliente quali nome, indirizzo e numero telefonico.

A seguito della creazione del file BAR nel modulo 330, si crea un record descrittore di sistema nel modulo 340. Un record descrittore di sistema, nella forma di attuazione preferita, è un file leggibile da elaboratore che è descrittivo dei componenti hardware e software da inserire all'interno del sistema di elaboratore obiettivo. In

una forma di attuazione preferita, il record descrittore di sistema contiene un elenco di componenti del sistema obiettivo in un formato comprendente etichette hardware, etichette software, etichette di informazioni e commenti. Una etichetta hardware identifica per il programma di sequenziamento 204 quelle informazioni che seguono l'etichetta come relative ad un componente hardware. Analogamente, l'etichetta software identifica informazioni che seguono l'etichetta come relative ad un componente software. L'etichetta di informazioni indica che devono seguire informazioni generali. I commenti consentono di inserire varie affermazioni nel record descrittore di sistema che sono ignorate dal programma di sequenziamento 204. Si desidera che il record descrittore di sistema sia un file di testo che sia leggibile da un uomo e facile da comprendere. Tale file consente in modo vantaggioso una risoluzione di errori e una manutenzione semplice del processo di installazione di test. Si potrà apprezzare che il record descrittore di sistema potrebbe essere qualsiasi elenco di identificatori unici che corrispondono ad un insieme unico di token per esempio, in un esempio il record descrittore di sistema può essere un elenco di numeri di particolare.

La Figura 3B illustra un ordine di sistema di elaboratore obiettivo 350 esemplificativo, un corrispondente file BAR 360 ed un corrispondente record descrittore di sistema 370. L'ordine di sistema di elaboratore obiettivo 350 contiene il nome di una famiglia di elaboratori, in questa illustrazione la famiglia "X". Compresi inoltre nell'ordine di sistema di elaboratore obiettivo 350 vi sono tre componenti hardware esemplificativi comprendenti un processore Pentium®, una unità a disco rigido, ed un monitor. Il file BAR 360 deriva dall'esecuzione dell'ordine di sistema di elaboratore obiettivo 350 attraverso un programma di conversione come illustrato nel modulo 320 di Figura 3A. Il file BAR 360 contiene un identificatore unico per il sistema di elaboratore obiettivo specifico all'interno della famiglia X. Il file BAR 360 comprende inoltre i numeri di particolare specifici per costruttore per ciascuno dei componenti elencati nell'ordine di sistema di elaboratore obiettivo. Inoltre, il file BAR 360 contiene un identificatore che indica la quantità desiderata di ogni componente così come una descrizione di testo di ogni componente da inserire sul sistema di elaboratore obiettivo. Il sistema 90 utilizza il file BAR 360 per creare il record descrittore di sistema 370.

Come illustrato, il record descrittore di sistema 370 contiene inoltre l'identificatore unico per il sistema di elaboratore obiettivo specifico all'interno della famiglia C. Inoltre, il record descrittore di sistema 370 contiene etichette appropriate, che indicano qui che il processore, l'unità a disco rigido e il monitor sono tutti componenti hardware, piuttosto che software. Il record descrittore di sistema 370 descrive i componenti in una descrizione di testo. In aggiunta, il record descrittore di sistema 370 esemplificativo contiene una etichetta software che indica che certo software deve essere installato o testato sul sistema di elaboratore obiettivo appartenente alla famiglia X. Per esempio, l'etichetta software potrebbe indicare che un certo sistema operativo appropriato per il processore Pentium® sia sempre installato sull'unità a disco rigido del sistema di elaboratore obiettivo appartenente alla famiglia X.

In Figura 4, è evidenziato il metodo generale preferito per sequenziare le fasi di installazione di software e di test. Nel modulo 400, l'identificatore unico del sistema di elaboratore obiettivo viene generato per il sistema di elaboratore obiettivo 160. Nella forma di attuazione illustrata in Figura 1, un utente che siede presso

il sistema di elaboratore a dispositivo di realizzazione di fase 140 fornisce l'identificatore unico (ad esempio l'identificatore BAR che funziona come codice di rintracciamento) nel programma di sequenziamento 204 del dispositivo di realizzazione di fase 140. In alternativa, nella forma di attuazione di Figura 2, l'identificatore unico viene letto automaticamente nel programma di sequenziamento 204 dopo che si riceve l'ordine di sistema di elaboratore obiettivo.

Nel modulo 410, si localizza un record descrittore di sistema corrispondente all'identificatore BAR. Nella forma di realizzazione di Figura 1, la connessione di rete 110 e la connessione di rete 195 localizzano il record descrittore di sistema. Nella forma di attuazione di Figura 2, la connessione di rete 195 localizza il record descrittore di sistema. Nel modulo 420, il record descrittore di sistema localizzato viene fornito al programma di sequenziamento 204. Nella forma di attuazione di Figura 1, il programma di sequenziamento richiede sul sistema di elaboratore al dispositivo di realizzazione di fase 140 mentre nella forma di attuazione di Figura 2, il programma di sequenziamento risiede sul server di file 202. Il programma di sequenziamento 204 opera insieme alla

banca dati 100 (delle Figure 1 e 2) per sequenziare le fasi di installazione di software e di test per il sistema di elaboratore obiettivo 160. Quando le fasi di installazione di software e di test appropriate per il particolare sistema di elaboratore obiettivo vengono sequenziate, il programma di sequenziamento 204 produce file di uscita come illustrato nel modulo 430.

Nella forma di attuazione illustrata in Figura 1, i file di uscita sono preferibilmente scritti sul disco di fase 150 (vedere Figura 1) in sei file separati. Questi file comprendono (1) un file di fasi, (2) un file Setenv.bat, (3) un file Qt.txt, (4) un file Et.txt, (5) un file Etlast.txt, e (6) un file Ft.txt. Si desidera che il file di fasi sia un file di testo ASCII comprendente un elenco di linee di comando appropriate per eseguire le fasi di installazione di software e di test per il sistema di elaboratore obiettivo ordinato. In una forma di attuazione preferita, il file di fase comprende inoltre comandi che possono essere eseguiti ad anello. Più specificamente, il file di fasi consente di ripetere i comandi per un numero definito di iterazioni o per una durata temporale definita. Tale formato consente in modo vantaggioso di ripetere le fasi di installazione di software o di test in

maniera calcolata e predeterminata. Il file Setenv.bat imposta preferibilmente variabili di ambiente sul sistema di elaboratore obiettivo. Si potrà apprezzare che in un modo di funzionamento, soltanto il file Step e il file Setenv.bat sono file ad istruzioni di testo ASCII contenenti un elenco di linee di comando appropriate per eseguire le fasi di installazione e di test per il sistema di elaboratore obiettivo. I file Qt.txt, Et.txt, Etlast.txt e Ft.txt sono preferibilmente tutti file di testo ASCII contenenti un elenco di linee di comando appropriate per eseguire diagnostica nelle fasi di fabbricazione di Test Rapido (Qt), di Test Esteso 1 (ET1), di Test Esteso 2 (ET2), di Installazione Software (SI) e di Test Finale (Ft) del sistema di elaboratore obiettivo.

Nella forma di attuazione di Figura 2, d'altra parte, i file di uscita non vengono scritti su un disco di fasi come illustrato in Figura 1. Invece, i file di uscita risiedono sul server di file 202 o sul server di file 190, sono utilizzati per dirigere l'esecuzione delle fasi di installazione di software e/o di test sul sistema di elaboratore obiettivo 160.

La Figura 5 illustra un diagramma schematico più dettagliato del funzionamento del programma di

sequenziamento 204 illustrato nelle Figure 1 e 2.

Il modulo di decisione 500 determina l'origine di un ordine. Per il momento, si consideri soltanto la diramazione sinistra. Se è richiesto, il modulo 502 applica uno o più patch al record descrittore di sistema. Nella forma di attuazione preferita, questo patch è modulare, consentendo di creare patch per un sistema di elaboratore obiettivo specifico, una particolare famiglia di sistemi di elaboratore o per un particolare componente. Per esempio, se un costruttore desiderasse sostituire una marca di unità a disco rigido con un'altra per una certa famiglia di sistemi di elaboratore in un certo giorno, si potrebbe formare un patch che modificherebbe tutti i record descrittori di sistema contenenti l'unità a disco rigido da sostituire e realizzare la sostituzione nel modulo 502.

Quindi, il modulo 504 invia in ingresso il record descrittore di sistema (con patch) corrispondente al sistema di elaboratore obiettivo 160 al programma di sequenziamento 204. Nel modulo 506, un descrittore di componenti viene letto dal record descrittore di sistema. Ogni descrittore di componente descrive un rispettivo componente, hardware o software, del sistema di elaboratore obiettivo.

Passando alla Figura 3B, la linea del record descrittore di sistema comprendente il processore Pentium® nel modulo 370 nel descrittore di corrente esemplificativo. Nel modulo 508, il programma di sequenziamento 204 esemplifica una pluralità di oggetti derivati corrispondenti rispettivamente alla pluralità dei componenti del sistema di elaboratore obiettivo 160. Nella forma di attuazione preferita, quegli oggetti derivati vengono utilizzati per memorizzare informazioni (ottenute dalla banca dati 100) relative alla fase di installazione di software e di test che occorre eseguire sul sistema di elaboratore obiettivo 160. Di conseguenza, nel modulo 510 ogni oggetto derivato è associato ad un rispettivo componente del sistema di elaboratore obiettivo 160.

A questo punto ci si riferisca alla diramazione destra dal modulo 550. In questo caso si ipotizza che l'ordine sia memorizzato direttamente da un cliente come record in una banca dati sotto forma di Distinta Base, tale record comprendendo descrittori di componente relativi al sistema di elaboratore obiettivo 160. Il modulo 512, equivalente al modulo 502, applica deviazioni (patch) alla Distinta Base mentre il modulo 514 legge la Distinta Base dalla banca dati su cui è memorizzata per l'utilizzo da

parte del programma di sequenziamento 204.

Nel modulo 516, le fasi di installazione di software o di testa associate ai rispettivi componenti del sistema di elaboratore obiettivo 160 vengono reperite dalla banca dati 100 e memorizzate nell'oggetto derivato appropriato. Nella forma di attuazione di Figura 1, le fasi vengono reperite tramite la connessione di rete 110, mentre nella forma di attuazione di Figura 2 le fasi vengono reperite direttamente dal server di file 202. Per descrivere come le fasi vengono reperite dalla banca dati 100 nella forma di attuazione preferita si richiede una descrizione della costruzione preferita di tale banca dati.

La Figura 6 illustra il progetto della banca dati 100. La banca dati 100 associa sequenze di fasi di installazione di software e/o di test, in un predeterminato ordine, con famiglie di sistemi di elaboratore. Inoltre, la banca dati 100 è configurata in modo da associare componenti di sistemi di elaboratore. Ancor di più, la banca dati 100 associa fasi di installazione di software e/o di test con componenti dei sistemi di elaboratore.

La banca dati 100 è preferibilmente una banca dati relazionale. La banca dati 100 contiene parecchie tabelle, contenenti ciascuna attributi

adatti a creare le associazioni sopra citate.

La banca dati 100 contiene la tabella Step 102, la tabella Family 104, la tabella FamilyStepSeq 106, la tabella Component 108, la tabella FamilyComponent 112, la tabella ComponentStep 114, la tabella StepDependency 116, la tabella StepParameter 118, la tabella ComponentClass 120, la tabella ComponentClassAttr 122 e la tabella OperatorMsg 124. Nella forma di attuazione preferita, ogni tabella contiene un elenco di attributi, gli attributi sottolineati servendo come chiave primaria.

La tabella Step 102 contiene tutte le fasi di installazione di software e di test per tutti i possibili componenti di tutte le famiglie di elaboratori. Nella costruzione preferita, la tabella Step 102 ha attributi comprendenti StepID, Name, Command, CommandType, AfterActionType, MaxIstance, ClassID e DepMask. StepID è un numero di identificazione unico per ogni fase di installazione di software o di test. Name è una stringa che assegna un nome che è descrittivo della fase. Command è una stringa che assegna una linea di comando eseguibile per eseguire la fase di installazione di software o di test sul sistema obiettivo 160 (illustrato nelle Figure 1 e 2). AfterActionType è un identificatore che determina se

necessario una fermata o una reinizializzazione dopo che si esegue la fase di installazione di software o di test. MaxIstance è un identificatore che indica il numero massimo di volte consentite per cui può essere eseguita la fase. ClassID identifica un certo tipo o classe di componente (ad esempio unità a disco rigido, unità CD-ROM) che è associato alla fase di installazione di software o di test. Infine, DepMask registra informazioni relative al fatto che una particolare fase abbia o meno una dipendenza di fase e/o un parametro di fase e pertanto determina se occorre entrare o meno nella tabella StepDependency 116 e/o nella tabella StepParameter 118.

La tabella Family 104 identifica ogni famiglia di sistemi di elaboratore per un intero di identificazione specificato nell'attributo FamilyID. Compresa inoltre nella tabella Family si ha una stringa che identifica il nome della famiglia.

La tabella FamilyStepSeq 106 è una tabella relazionale che contiene la relazione tra la tabella Step 102 e la tabella Family 106. La tabella FamilyStepSeq 106 comprende un intero di identificazione di famiglia specificato nell'attributo FamilyID per una particolare famiglia di sistemi di elaboratore (dalla tabella Family

104), un intero di identificazione di fase specificato nell'attributo StepID (dalla tabella Step 102) che identifica un particolare insieme di fasi appropriate per tali famiglie, ed un numero di sequenza. Il numero di sequenza è contenuto all'interno dell'attributo StepSeqNum che rappresenta un predeterminato ordine in cui devono essere eseguite le fasi associate ad una particolare famiglia. Ingegneri di test assegnano numeri di sequenza, unici nell'ambito di ogni fase di fabbricazione, in un ordine scelto in modo da essere il più efficace per un particolare sistema obiettivo. Si potrà apprezzare che si possono utilizzare altri modi per assegnare numeri di sequenza. Infine, la tabella FamilyStepSeq 106 comprende PhaseID dalla tabella Step 102.

La tabella Component 108 contiene tutti i possibili componenti che sono compresi all'interno dei sistemi di elaboratore in fabbricazione. Gli attributi di questa tabella sono ComponentID che assegna un identificatore ad ogni componente, Description che assegna un nome di stringa ad ogni componente, e ClassID che fa riferimento al tipo di componente (ad esempio unità a disco rigido, unità CD-ROM).

La tabella FamilyComponent 112 è una tabella

relazionale che contiene relazioni tra ogni famiglia di sistemi di elaboratore ed un insieme di componenti che possono essere inseriti in tale famiglia. Gli attributi della tabella Family Component 112 comprendono un intero di identificazione di famiglia di elaboratori specificato nell'attributo FamilyID (dalla tabella Family 104) ed un intero di identificazione di componente specificato nell'attributo FamilyID (dalla tabella Component 108).

La tabella ComponentStep 114 è una tabella relazionale contenente relazioni tra ogni componente ad un insieme di fasi di installazione di software e di test appropriate per tale componente. Gli attributi della tabella ComponentStep 114 comprendono un intero di identificazione di componente specificato nell'attributo ComponentID (dalla tabella Component 108) ed un intero di identificazione di fase specificato nell'attributo StepID (dalla tabella Step 102).

La tabella StepDependency 116 contiene dati relativi a possibili conflitti. Certi test possono entrare in conflitto con certe classi di componente, o i componenti specifici stessi o componenti da certi costruttori. Per esempio, il sistema di elaboratore obiettivo 160 da costruire può

comprendere una unità a disco rigido di marca A ed un CD-ROM di marca B. L'unità a disco rigido di marca A può richiedere comunemente l'esecuzione del test C ma può risultare che il test C sia incompatibile con l'unità CD-ROM B; tutte queste dipendenze sono registrate nella tabella 116. In questa tabella, StepID identifica la fase avente una dipendenza, TypeID indica se la dipendenza è o meno rispetto ad una classe di componenti o un componente specifico, ObjectID è un ClassID oppure un ComponentID a seconda dello stato di TypeID, e DepTypeID indica se una particolare fase deve essere tenuta o tolta e quando si verifica un conflitto.

La tabella StepParameter 118 identifica parametri che certe fasi possono richiedere; per esempio, si può richiedere che una fase sia in esecuzione per una lunghezza di tempo specifica, o sia in esecuzione attraverso un numero specifico di iterazioni. Nella tabella 118, StepID identifica in modo unico la particolare fase di installazione di test. ParameterID identifica ogni parametro associato a tale fase; ci può essere più di un parametro associato ad una particolare fase e ciascuno avrà il proprio ParameterID. Per esempio, si può utilizzare lo stesso test, ma con parametri diversi, per marche diverse di unità a disco rigido.

DataType identifica il tipo di dati che deve essere inserito nel rispettivo parametro. Nel suddetto esempio, il DataType può specificare che i dati sono relativi ad una percentuale o, in alternativa, ad un codice di ID di una unità a disco rigido. Content è un elemento di commutazione di linea di comando come utilizzato nel linguaggio di programmazione C in associazione ad un comando come "printf". Per esempio, Content può essere "-%d" per indicare che una percentuale è appropriata per questo parametro. StepSeqNum e ClassID sono come descritto in precedenza.

Occorre notare che la tabella StepParameter 118 memorizza soltanto i tipi e il numero di parametri associati ad una particolare fase, ma di fatto non memorizza i valori di questi parametri. Così, durante la costruzione di un file di fasi che deve essere riscritto, la tabella 118 non inserisce valori di parametro in una linea di comandi del file di fasi. Piuttosto, essa contiene tutti i dettagli necessari a consentire la costruzione della linea di comando. E' il programma di sequenziamento 204 che calcola il valore del parametro ed inserisce lo stesso nella linea di comando di file di fasi durante l'esecuzione. Il programma di sequenziamento esegue il calcolo sulla base delle informazioni che

sono contenute nel record descrittore.

Il vantaggio di avere la tabella StepParameter 118 è quello di consentire una maggior flessibilità evitando la necessità di avere parametri associati in permanenza alle fasi. Così la tabella 118 consente ad un ingegnere di modificare immediatamente i parametri senza dover modificare la tabella Step 102.

La tabella ComponentClass 120 è semplicemente un elenco di tutte le classi di componenti (ClassID), ad esempio unità a disco rigido, CD-ROM, etc., insieme ad una breve descrizione di queste classi (ClassName, Description).

La tabella ComponentClassAttr 122 elenca tutte le classi e tutti gli attributi associati ad ogni classe. AttrID è un codice assegnato ad ogni tipo diverso di attributo quale dimensione di memoria, velocità operativa, costruttore, ecc., mentre AttrName è un nome più descrittivo dell'attributo a beneficio di un ingegnere. DataType è una indicazione del tipo di dati che sono utilizzati per rappresentare un particolare attributo. Per esempio, ci possono essere una stringa di caratteri nel caso in cui l'attributo sia il nome del costruttore, oppure possono essere un intero sull'attributo della dimensione di memoria.

La tabella ComponentClass 120 e la tabella ComponentClassAttr 122 non sono utilizzate attivamente dal programma di sequenziamento 204, ma sono utilizzate principalmente dagli ingegneri di sviluppo. Queste tabelle non comprendono alcun valore effettivo degli attributi.

Infine, la tabella OperatorMsg 124 memorizza un certo numero di messaggi per l'operatore di test a seconda del test eseguito e i componenti testati. Per esempio, una richiesta può essere emessa per ricordare ad un operatore di collocare un nastro in una unità a nastro prima di testare l'unità a nastro.

Il sistema di elaboratore obiettivo esemplificato illustrato in Figura 3B sarà utilizzato per illustrare come si impiega il progetto di banca dati evidenziato in precedenza per reperire le fasi di installazione di software e di test. L'identificatore di famiglia di elaboratore nel record descrittore di sistema che identifica la famiglia X è associato al FamilyID corrispondente alla famiglia X nella tabella Family 104. Si utilizza la tabella Component 108 per controllare se i componenti del sistema di elaboratore obiettivo elencati nell'ordine di sistema elaboratore obiettivo sono legali. In altre parole, il programma

di sequenziamento e la banca dati determinano se il processore, l'unità a disco rigido, il monitor e i software contenuti nel record descrittore di sistema Figura 3B hanno corrispondenti inserimenti e corrispondenti interi specificati da ComponentID nella tabella Component 108. Se un componente non è legale (cioè se un componente nel record descrittore di sistema non è contenuto nella tabella Component 108), si crea un indicatore di errore. La tabella FamilyComponent 112 è una tabella relazionale che contiene mappature dalla tabella Component 108 e dalla tabella Family 104. La tabella FamilyComponent 112 contiene tutti i componenti legali che possono essere compresi su un sistema di elaboratore obiettivo appartenente alla famiglia X. Così, la tabella FamilyComponent 112 può essere utilizzata per controllare se tutti i componenti del sistema obiettivo sono legali. In altre parole, il programma di sequenziamento e la banca dati determinano se il processore, l'unità a disco rigido, il monitor e il software contenuti nel record descrittore di sistema di Figura 3B hanno corrispondenti relazioni nella tabella FamilyComponent 112. Se un componente non è legale (cioè se un componente nel record descrittore di sistema può non essere compreso su un sistema obiettivo appartenente alla famiglia X), si crea un

indicatore di errore.

Nella tabella relazionale FamilyStepSeq 106 risiedono mappature dalla tabella Step 102 e dalla tabella Family 104. La tabella FamilyStepSeq 106 contiene tutte le fasi di installazione di software e di test che possono essere eseguite legalmente sui sistemi di elaboratore obiettivo appartenenti alla famiglia X. Inoltre, è in questa tabella FamilyStepSeq 106 che i numeri di sequenza di fase sono associati ad ogni fase di installazione di software di test. Questi numeri di sequenza di fase rappresentano l'ordine appropriato in cui devono essere eseguite le fasi per una particolare famiglia di sistemi di elaboratore. Pertanto, la tabella FamilyStepSeq 106 contiene un listato di fasi che devono essere eseguite su sistemi elaboratore obiettivo della famiglia X così come numeri di sequenza di fase che rappresentano un predeterminato ordine in cui devono essere eseguite le fasi.

La tabella ComponentStep 114 è una tabella relazionale che contiene mappature dalla tabella Component 108 e dalla tabella Step 102. La tabella ComponentStep 114 contiene le fasi di installazione di software e di test da eseguire per il processore, l'unità a disco rigido, il monitor e il software del sistema di elaboratore obiettivo.

Il reperimento delle fasi di installazione di software e di test associate ai rispettivi componenti da inserire sul sistema obiettivo comporta l'esecuzione di una operazione di unione sulla tabella FamilyComponent 112 e la tabella ComponentStep 114 per ottenere un insieme intermedio che elenca le fasi da eseguire sui componenti del sistema di elaboratore obiettivo 160.

L'operazione di unione dà come risultato un elenco di fasi da eseguire sul processore, sull'unità a disco rigido, sul monitor, e sul software elencate nel record descrittore di sistema illustrato in Figura 3B. Il risultato dell'unione e della tabella FamilyComponent 112 e della tabella ComponentStep 114 viene quindi unito alla tabella FamilyStepSeq 106 che contiene tutte le fasi per la famiglia X. Il risultato di questa operazione di unione comprende informazioni di sequenziamento sotto forma di numeri di sequenza e numeri di fase, i numeri di sequenza essendo unici all'interno di una particolare fase. Così, una unione di tre tabelle della tabella FamilyComponent 112, della tabella ComponentStep 114, e della tabella FamilyStepSeq 106 produce le fasi di installazione di software e di testa appropriate così come le informazioni di sequenziamento sotto forma di numeri

di sequenze di fase per installare e/o testare software sul sistema di elaboratore obiettivo 160.

Se il risultato della prima operazione di unione (l'unione della tabella FamilyComponent 112 e della tabella ComponentStep 114) è un insieme vuoto, occorre creare una condizione di errore, per un insieme vuoto che segnali che il componente da inserire sul sistema obiettivo non appartiene alla famiglia elencata sul record descrittore di sistema. Un esempio di ciò è illustrativo. Si consideri che un record descrittore di sistema indichi correttamente che un sistema di elaboratore obiettivo appartiene alla famiglia Y. Si ipotizzi, tuttavia, che il record descrittore di sistema indichi in modo non corretto che una unità a disco rigido (unità a disco rigido Z) appartenente soltanto ai sistemi obiettivo nella famiglia X deve essere inserita sul sistema obiettivo che si trova nella famiglia Y. In tal caso, la tabella ComponentStep 114 contiene fasi associate all'unità a disco rigido Z. La tabella FamilyComponent 112 contiene componenti associati alla famiglia Y. Così, l'unione della tabella ComponentStep 114 con la tabella FamilyComponent 112 produce un insieme vuoto, dato che l'unità a disco rigido Z non è componente associato alla famiglia Y (invece, esso è

associato soltanto alla famiglia X). Come appare evidente dal suddetto esempio, il processo preferito della banca dati consente in modo vantaggioso di accertarsi che un sistema obiettivo di una certa famiglia contenga soltanto componenti appropriati per tale famiglia.

Con riferimento di nuovo alla Figura 5, dopo che si reperiscono le fasi associate ai componenti da inserire nel sistema obiettivo, il modulo 518 del programma di sequenziamento 204 determina, per ogni fase, se esiste una dipendenza di fasi esaminando DepMask per tale fase. Se è così, il modulo 520 legge la dipendenza dalla tabella StepDependency 116 e il modulo 522 risolve la dipendenza secondo DepTypeID.

Quindi, il modulo 524 determina se la fase richiede un parametro, di nuovo esaminando DepMask per tale fase. Se è così, il modulo 526 legge i dati di parametro dalla tabella StepParameter 118 e il modulo 528 calcola il valore effettivo del parametro e lo inserisce nella linea di comando della fase.

Ora, il modulo 530 prepara le variabili di ambiente per il sistema di elaboratore obiettivo leggendo il record descrittore di sistema e creando un file di ambiente corrispondente ai componenti da inserire sul sistema obiettivo. Per esempio, si

legge il record descrittore di sistema illustrato in Figura 3D, e si potrebbe preparare una variabile di ambiente quale "set cpu=pentium" corrispondente al componente hardware di processore del record descritto nel sistema.

Nel modulo 532, la pluralità di fasi di installazione di software e di test, reperite tramite l'unione di tre tabelle sopra descritte e con le dipendenze risolte ai parametri aggiunti, vengono sequenziate nell'ordine predeterminato. Questo sequenziamento avviene secondo i rispettivi numeri di sequenza e i numeri di fase per fornire una sequenza di fasi. Il sequenziamento stesso può essere realizzato utilizzando uno qualsiasi di molti algoritmi di ordinamento ben noti nella tecnica.

Nel modulo 534, il programma di sequenziamento 204 invia in uscita file. Questi file comprendono (1) un file di fasi, (2) un file Setenv.bat, (3) un file Qt.txt, (4) un file Et.txt, (5) un file Etlast.txt e (6) un file Ft.txt. Si desidera che il file di fasi sia un file di testo ASCII. In una forma di attuazione preferita, il file di fasi consente di ripetere i comandi per un numero definito di iterazioni o per una durata temporale definita. Il file Setenv.bat imposta le variabili di ambiente sul sistema di elaboratore obiettivo. Il

file di fasi comprende le fasi che devono essere eseguite rispettivamente durante le fasi di fabbricazione Test Rapido (Qt), Test Esteso 1 (ET1), Test Esteso 2 (ET2), Installazione Software (SI) e Test Finale (FT) del sistema di elaboratore obiettivo.

Come illustrato, per la forma di attuazione di Figura 2, il modulo 534 memorizza i file di uscita, come tali o in un banca dati, sul server di file 202. I file di uscita scritti sul server di file 202 possono essere utilizzati per dirigere l'esecuzione delle fasi di installazione test di software sul sistema di elaboratore obiettivo 160.

Nel modulo 536, la sequenza di fasi, se richieste, è modificata utilizzando un patch di sequenza di fasi. Nella forma di attuazione preferita, questo patch è modulare, consentendo la creazione di patch per un sistema di elaboratore obiettivo specifico, una particolare famiglia di sistemi di elaboratore o per un particolare componente. Per esempio, se un costruttore desiderasse eseguire una fase di test prima di un'altra per un certo componente in un certo giorno, si potrebbe formare un patch che modificherebbe tutte le sequenze di fase contenente le fasi in cui l'ordine deve essere modificato e non cambierebbe in

modo corrispondente l'ordine di esecuzione nel modulo 536. A seguito dei patch, il modulo 538 invia in uscita file revisionati per la memorizzazione, di nuovo come tali, oppure in una banca dati, sul server di file 202.

Infine, il modulo 540 fornisce l'opzione di scrivere su un dischetto 150, Figura 1. Se è richiesto un dischetto, invece di scrivere direttamente sul dischetto, il modulo 542 crea un "dischetto virtuale" in memoria e quindi il modulo 544 scrive l'intero dischetto virtuale sul dischetto fisico in una operazione e ciò riduce il numero di operazioni di scrittura su una unità a dischi floppy e velocizza in tal modo significativamente il funzionamento globale del programma.

Il dischetto virtuale viene creato dal seguente programma che crea un equivalente di memoria del dischetto fisico allocando un insieme di blocchi di memoria equivalenti ciascuno alla dimensione di un settore fisico sul dischetto fisico. Il sistema di file è FAT 12 (utilizzato dai sistemi operativi PC-DOS, MS-DOS, Windows 95 e Windows NT). Il primo settore è il settore di inizializzazione del dischetto. Un gruppo è un raggruppamento/unità logica di un insieme di settori. Questo numero è fisso una volta che si inizializza un sistema di

file. Per esempio, una dimensione di gruppo è due settori. Il sistema di file consente soltanto la locazione per gruppo e non per settore. In questo caso, il file più piccolo consumerà almeno un gruppo (o 2 settori).

Inizio

Creare un insieme di blocchi di memoria. Numero di blocco di memoria equivalente al numero di settore finito sul dischetto con il dato sistema di file.

Inizializzare tutto il contenuto dei blocchi di memoria a zero

Inizializzare il settore di inizializzazione copiando una immagine esterna del solo settore di inizializzazione. Questa immagine esterna è memorizzata in un file.

Inizializzare la tabella di FAT. (Settori ben definiti sul dischetto tramite il sistema di file).

Se si richiede l'operazione di scrittura di file.

Leggere il file

Allocare i gruppi necessari

Se si ha un errore, uscire la funzione con un errore poiché non si ha spazio sufficiente.

Aggiornare la directory e la tabella di FAT per i gruppi allocati.

Scrivere il contenuto letto sui gruppi

Se si richiede l'operazione di cancellazione di file.

Allocare liberamente i gruppi per il dato file

Aggiornare la directory e la tabella di FAT per i gruppi liberati.

Se si richiede una operazione di scrittura di dischetto fisico.

Ottenere il conteggio di utilizzo di dischetto memorizzato nel quarto byte del settore di inizializzazione dal dischetto.

Se il conteggio è $\geq$ del conteggio massimo, restituire un codice di errore per indicare la regione dell'errore.

Se il conteggio è < del conteggio massimo, incrementare il conteggio di 1.

Riscrivere il valore di conteggio nel terzo byte del settore di inizializzazione sul dischetto virtuale.

Scrivere i blocchi di memoria dal dischetto virtuale al dischetto fisico, fermarsi quando non è rimasto alcun blocco di memoria ulteriore che contiene dati.

Fine.

Passando di nuovo alle Figure 1 e 2, la freccia 130 illustra il fatto che si possono effettuare modifiche alla banca dati 100. Per esempio, se si crea una nuova famiglia di sistemi di elaboratore, si può modificare la banca dati 100 di conseguenza. Più specificamente, si assegna alla nuova famiglia un nuovo identificatore di famiglia in FamilyID della tabella Family 104 e si assegna un nome per la nuova famiglia all'attributo Name della tabella 104. Si aggiunge un elenco di fasi di installazione di software e fasi di test alla tabella FamilyStepSeq 106, queste fasi rappresentando quali fasi devono essere eseguite, e in quale predeterminato ordine, sulla nuova famiglia di sistemi di elaboratore. La nuova di sistemi di elaboratore condivide parecchie somiglianze con una famiglia esistente, è probabile che si possono modificare i riferimenti per la famiglia esistente nella tabella FamilyStepSeq per produrre riferimenti per la nuova famiglia. Se occorre creare qualche nuova fase per la nuova famiglia di sistemi di elaboratore, queste fasi vengono aggiunte alla tabella Step 102. Analogamente, se qualche nuovo componente accompagna la nuova famiglia di sistemi di elaboratore, questi componenti vengono aggiunti alla tabella Component

108. La tabella ComponentStep 114 è aggiornata per associare ogni componente della nuova famiglia di sistemi di elaboratore con le fasi appropriate per la sua installazione di software e di test. Se la nuova famiglia utilizza soltanto componenti già presenti nella banca dati, non occorre modificare questa tabella. La tabella FamilyComponent 112 è aggiornata in modo tale che un elenco di componenti consentiti che possono essere inseriti sulla nuova famiglia si troverebbe nella banca dati. In particolare, sarebbe necessario associare il SysID nel nuovo sistema di elaboratore con il CompID di ogni componente consentito. Di nuovo, questo può essere effettuato copiando e quindi modificando un inserimento esistente di una famiglia più vecchia di sistemi di elaboratore.

Si potrà apprezzare che nella costruzione di una banca dati secondo la forma di attuazione preferita, sono previsti certi vantaggi significativi. In particolare, il progetto modulare della banca dati consente in modo vantaggioso una impostazione semplice e delle fasi di installazione di software e di test per nuove famiglie di sistemi di elaboratore. In aggiunta, le fasi di installazioni di software e di test per una particolare famiglia di sistemi di elaboratore o per

un particolare componente possono essere modificati indipendentemente da altre fasi di installazione di software e di test.

Si presterà ora attenzione all'esecuzione della sequenza di fasi sul sistema obiettivo 160. Le fasi di installazione di software e di test sono eseguite sul sistema di elaboratore obiettivo 160 utilizzando un programma che legge, interpreta ed esegue la sequenza di fasi corrispondente al sistema di elaboratore obiettivo. Nella forma di attuazione preferita questo programma è chiamato RunStep ed è collocato sul disco di fasi 150 nella forma di attuazione di Figura 1 e sul server di file 202 nella forma di attuazione di Figura 2.

La Figura 7 illustra una porzione di una sequenza di fasi contenuta in un file di fasi prima che si sia eseguita qualsiasi fase di installazione di software e di test. Come citato in precedenza, la sequenza di fasi comprende comandi per installare software e/o per testare il sistema di elaboratore obiettivo costruito secondo un ordine. In aggiunta, la sequenza di fasi nel file di fasi consente di ripetere i comandi per un numero definito di iterazioni o per una durata temporale definita. Inoltre, il file di fasi può contenere commenti, ignorati dal programma RunStep. Nel file di fasi, si

utilizzano segni 800 per separare campi della sequenza di fasi. Le voci 810 sono comandi per testare il sistema di elaboratore obiettivo 160. I comandi comprendono, per esempio, un comando per testare la memoria e per testare dispositivi di interfaccia di sistemi di elaboratori piccoli (SCSI). Come si può vedere dalla figura, ogni comando può comprendere interruttori quali "-o" appropriati per il particolare ambiente di test. La voce 820 è un commento che è ignorato dal programma RunStep. La voce 810c è un comando che viene eseguito ad anello nel tempo. Nella costruzione preferita, l'istruzione 'begin_time_loop" designa il punto di partenza di un anello. L'istruzione 'end_time_loop" designa il punto terminale di un anello. L'istruzione 'end_time_loop' si combina con il campo che designa la durata temporale di iterazione attraverso l'anello. Qui, per esempio, il comando 810c è eseguito per un'ora e trenta minuti. La voce 810d ha un comando che viene eseguito ad anello secondo un certo numero di iterazioni. Nella forma di attuazione preferita, il comando 'begin_iterate_loop' istruisce il programma RunStep all'esecuzione di un anello iterattivo. Il comando 'end_iterate_loop' segnala la fine dei comandi di esecuzione ad anello. Qui, il comando 810d è

eseguito tre volte.

Le Figure da 8 a 13 illustrano il diagramma di flusso per il programma RunStep. Come panoramica, RunStep elabora ogni file di fase una linea per volta piuttosto che leggere l'intero file di fase in memoria. In ogni linea RunStep esegue un certo numero di controlli per valutare se deve continuare o meno ad elaborare tale linea. Per esempio, se RunStep vede che si è registrata una condizione di guasto da quando si era eseguita la linea precedente, esso sa che non ha scopo continuare con il programma. In alternativa RunStep può controllare per vedere se un operatore ha manipolato o meno il file di fase (allo scopo di saltare per esempio un test noioso) e se è così non continuerà con il programma costringendo in tal modo l'operatore a cominciare dall'inizio. Così, una particolare riga di un file di fasi viene letta soltanto se RunStep determina che tale riga è in modo appropriato quella che deve essere eseguita successivamente - pertanto non si ha alcuna lettura non necessaria di righe dal file di fasi. Chiaramente, questa è una caratteristica che fa risparmiare tempo.

La Figura 8 è un diagramma di flusso di livello alto di RunStep. Il primo modulo 900 inizializza lo stato del sistema. Questo viene eseguito prima di

leggere qualsiasi linea di un file di fase. Durante questo stadio, RunStep legge le varie variabili di ambiente (da un file "progress.bat", che sarà descritto) in modo tale da sapere lo stato preciso del sistema, ad esempio se era stato restituito un guasto nell'esecuzione dell'ultima linea, se deve essere eseguita una riesecuzione, oppure RunStep può procedere con la lettura della linea successiva.

La Figura 9 descrive lo stadio di inizializzazione in maggior dettaglio. Il primo modulo 902 disabilita l'interruzione di controllo - questo impedisce ad un operatore di interrompere il programma per saltare una fase. Si inizializzano quindi le variabili, il modulo 904, e il programma legge le variabili ambientali dall'ambiente, modulo 906. Queste variabili ambientali sono memorizzate principalmente in un file batch chiamato progress.bat che è mantenuto in memoria e nel quale RunStep avrà scritto quando stava eseguendo linee precedenti nel file di fasi. Progress.bat contiene lo stato corrente del sistema, e come sarà descritto (modulo 106 Figura 13) viene aggiornato per ogni linea letta dal file di fasi. Le variabili di ambiente comprenderanno informazioni quali il numero di linea dell'ultima fase eseguita; l'identificazione di quale comando effettivo è stato

eseguito; quanto tempo è trascorso se si esegue un particolare comando all'interno di un anello temporale; quale fase di testo di installazione di software viene eseguita.

Se le variabili ambientali vengono lette positivamente, come determinato dal modulo 908, il modulo 910 legge l'elenco di directory di tutti i file sull'unità locale nella memoria dell'elaboratore 160 o 202 su cui è in esecuzione RunStep. Così quando RunStep arriva a controllare se un file si trova o meno sull'unità locale (modulo 1200 Figura 13), esso può farlo leggendo l'elenco di directory in memoria e non deve ricercare l'unità locale stessa. Questo risparmia tempo. Se RunStep ha successo nella lettura dell'elenco di directory, come determinato dal modulo 912, il modulo 914 ottiene lo stato di processo corrente. Durante questo stadio, RunStep imposta un certo numero di indicatori secondo lo stato del sistema, cioè guasto, riesecuzione, ecc. Sono questi indicatori che determineranno il flusso di RunStep attraverso successivi moduli 916, 918, 920. Questi moduli dirigeranno RunStep nella subroutine A, B o C, Figura 10, come applicabile.

Con riferimento alla Figura 10(a), si entra nella routine A se RunStep è stabilito nel modulo

916 della fase successiva alla normale fase di processo. Il primo modulo nella routine A, modulo 922, verifica varie somme di controllo. Lo scopo di ciò è garantire che non ci sia stata alcuna manipolazione con il file di fase, ad esempio RunStep legge certe somme di controllo e le correla con le somme di controllo che sono memorizzate in progress.bat - qualsiasi discrepanza indicherebbe che una persona non autorizzata ha modificato il file di fase. Se tutte le somme di controllo sono corrette, come determinato dal modulo 924, il programma azzera (cancella) certe variabili e legge l'ora corrente, modulo 926.

Rientra nella routine B, Figura 10(b), se l'ambiente indica che occorre rieseguire l'ultima fase, come determinato dal modulo 918. Lo scopo della routine B è quello di fornire ad un operatore non autorizzato una opportunità di interrompere il programma per evitare di rieseguire l'ultima fase se lo si desidera. Pertanto, nel modulo 928 viene visualizzato un messaggio che l'operatore può selezionare gli Strumenti di Produzione (MFGTools) entro cinque secondi oppure si avrà la riesecuzione. Strumenti di Produzione è un metodo approvato per cui un operatore autorizzato con la particolare chiave appropriata può irrompere nel file di fasi e

modificarlo come desidera. Il modulo 930 determina se deve essere rieseguita l'ultima fase, a seconda della risposta dell'operatore, e se è così il modulo 932 imposta l'ambiente di progresso (progress.bat) ad eseguire MFGTools.

La routine C, Figura 10(c) imposta semplicemente l'ambiente di progresso (progress.bat) ad eseguire MFGTools, modulo 934.

Ritornando alla Figura 9, il modulo 936 determina se è stato restituito un guasto, e se non è così, il modulo 938 controlla per vedere se la fase corrente è legale.

Con riferimento di nuovo alla Figura 8, quando si completa il modulo di inizializzazione 900, e se era stato restituito "successo" come determinato dal modulo 950, allora RunStep procede al modulo 952, che è "elaborare il file di fasi". Nel tempo in cui RunStep ha raggiunto questo stadio, essa ha letto informazioni sufficienti dall'ambiente per sapere se deve procedere o meno con la linea successiva del file di fasi, rieseguire la linea precedente, o interrompere.

La Figura 11 illustra il modulo 952, "elaborare il file di fasi", in maggior dettaglio. Il modulo 954 stabilisce se è stato selezionato o meno gli Strumenti di Produzione o se è richiesta una

riesecuzione. Se uno di questi è vero non è necessaria alcuna ulteriore impostazione e questa parte del programma può restituire un successo. Se nessuna di queste condizioni è vera (cioè se si suppone che RunStep si porti ora alla linea successiva del file di fasi), si apre il file di fasi della fase appropriata nel modulo 956. Ciò significa che RunStep apre il file di fasi associato alla particolare fase di test o installazione di software che viene condotta (cioè test rapido, test esteso, ecc.).

Il modulo 958 stabilisce se esiste o meno una fase di riavvolgimento. Una fase di riavvolgimento è il caso speciale di una riesecuzione in cui occorre ripetere l'intera fase di test nel caso di un guasto di una delle fasi - cioè RunStep deve ritornare all'inizio del file di fasi per tale fase. Se RunStep stabilisce che essa è una fase di riavvolgimento, il modulo 960 imposta l'ambiente di conseguenza e si restituisce un successo.

Se essa non è una fase di riavvolgimento, il modulo 962 legge il successivo numero di linea dal file di fasi e controlla inoltre che il numero di linea che ha appena letto sia la linea che esso si attende di leggere facendo corrispondere il numero di linea con quello memorizzato in progress.bat che

RunStep ha letto durante lo stadio di inizializzazione - di nuovo questo è un dispositivo anti-manipolazioni. In particolare, quando RunStep è in esecuzione, progress.bat conterrà informazioni relative all'ultima fase che era stata eseguita, cioè informazioni quali il numero di linea nel file di fasi, il ID di comando del comando che era stato eseguito, e altri controlli e informazioni rilevanti. Nel modulo 962 RunStep legge il file progress.bat contenuto in memoria, e salta in giù un numero specifico di linee del file di fasi come determinato dal numero di linee in progress.bat. Se il numero di linea a cui si trova attualmente corrisponde o meno al numero di linea a cui esso si aspetta di trovarsi. Esso controlla inoltre se il comando in questa linea è lo stesso o meno di quello che gli è stato detto essere stato appena eseguito, cioè RunStep convalida il fatto che la linea in cui ora si trova di fatto è l'ultima fase che era stata eseguita.

Se è così, come determinato dal modulo 964, nel modulo 966 RunStep legge la fase e imposta l'ambiente di progresso per eseguire la fase successiva. Il modulo 968 controlla che la fase letta fosse valida e se è così restituisce un successo. Se non si era letta una fase valida,

RunStep controlla il modulo 970 per vedere se esso ha di fatto raggiunto o meno l'ultima linea del file di fasi dell'ultima fase - se è così, RunStep restituisce un messaggio di "tutte fasi elaborate" indicativo che tutti i test sono completi. Se non è così, RunStep restituisce un guasto.

Se, d'altra parte, il modulo 964 di RunStep dimostra che il numero di linea che ha appena letto non corrisponde al numero di linea che stava aspettando, RunStep procede alla routine illustrata in Figura 12. Il primo modulo 972 controlla se RunStep ha di fatto raggiunto l'ultima linea del file di fasi per l'ultima fase. Se è così, viene restituito un messaggio che indica che tutti i test sono completi. Se non è così, il modulo 974 controlla se l'ultima linea in file di fasi è stata raggiunta. Se è così, RunStep ritorna al modulo 956 e apre il file di fasi dalla fase di test successiva.

Se, d'altra parte, RunStep stabilisce che non è la fine del file di fasi, esso controlla nel modulo 976 per vedere se il numero di linea che appena letto supera o meno la linea che stava aspettando (dalla lettura dalla memoria). Se il numero di linee è superato, questo indica a RunStep che si è avuta una manipolazione manuale del file di fasi (ad

esempio un operatore ha tolto una fase dal file di fasi) e RunStep restituisce un guasto. Se il numero di linea non è superato, RunStep ritorna al modulo 962.

RunStep ritorna quindi al modulo 990, Figura 8, e se si è inquadrata la fine del file di fasi, esso lo segnala all'operatore. RunStep controlla inoltre se è stato restituito o meno un errore sul modulo 992. Se è così, questo viene segnalato e si esce da RunStep.

In questo stadio, RunStep ha determinato lo stato preciso del sistema, sa se sta per rieseguire o meno una fase, riavvolge una fase o eseguire la linea successiva nel file di fasi e ha impostato la memoria di conseguenza. Prima di continuare, RunStep salva tutte le informazioni che ha preso negli stadi precedenti. Questo viene fatto nel modulo 994 che è illustrato in maggior dettaglio in Figura 13.

Prima di tutto, il modulo 1000 ha l'opportunità di cancellare qualsiasi file che non è più necessario. Quindi, il modulo 1002 determina se i file necessari per i test o l'installazione di software che stanno per aver luogo sono memorizzati o meno sull'unità locale o devono essere ottenuti su una rete, e salva queste informazioni. Quindi, modulo 1004, 1006, RunStep utilizza le informazioni

che ha appreso negli stadi precedenti, per impostare le variabili di ambiente (in progress.bat) cosicché la volta successiva che si esegue RunStep dall'inizio, tutte le variabili di ambiente sono state aggiornate per rappresentare lo stato attuale del sistema.

Dopo aver controllato nel modulo 1008 che la struttura è stata positiva, RunStep scrive quindi un file di registro nel modulo 1010. Se questo è positivo, il modulo 1012, si determina la necessità di una riesecuzione nel modulo 1014 e se ne è necessaria una il modulo 1016 crea un file di riesecuzione.

Il controllo ritorna quindi al modulo 1020, Figura 8, e se non era stato restituito alcun errore, si esce dal programma con livello di errore 255 che indica che dovrebbe aver luogo l'esecuzione della linea di comando letta dal file di fasi nel modulo 966. Ciò avviene nel modulo 1022. Quindi RunStep ritorna all'inizio per elaborare la fase successiva (linea nel file di fasi).

Si vedrà che il programma RunStep è un sistema ad alta sicurezza per il fatto che esistono vari controlli compresi per impedire ad un operatore non autorizzato di manipolare il file di fasi allo scopo di eliminare test noiosi o evitare riesecuzioni. Si

realizza ciò disabilitando l'interruzione di controllo, verificando i numeri di linea del file di fasi in vari intervalli; e collocando somme di controllo nel sistema. Un altro aspetto di sicurezza è che ogni qualvolta si verifica un errore, come determinato da RunStep in vari punti nel suddetto diagramma di flusso, si esce da RunStep e si scrive l'errore in un file nascosto di sola lettura; un operatore casuale non sarebbe conscio del file e non sarebbe in grado di localizzarlo e non potrebbe pertanto comprendere che cosa non è andato bene e come saltare l'errore, cioè sarebbe costretto a scartare il componente o rieseguire il testo fino a quando è soddisfacente.

Anche se sono state illustrate e descritte forme di attuazione illustrative, un'ampia gamma di modifiche, variazioni e sostituzioni è contemplata nella precedente descrizione e in alcuni esempi, si possono impiegare alcune caratteristiche delle forme di attuazione senza un utilizzo corrispondente delle altre caratteristiche. Di conseguenza, è appropriato che le rivendicazioni allegate siano costruite in modo ampio e in maniera coerente con il campo di protezione delle forme di attuazione qui descritte.

RIVENDICAZIONI

1. - Procedimento per l'installazione di software su un sistema di elaboratore, comprendente le fasi di prevedere una sequenza di fasi comprendente una pluralità di fasi di installazione di software da eseguire in un ordine determinato dalla sequenza di fasi; e

leggere ed eseguire fasi consecutive dalla sequenza di fasi.

2. - Procedimento secondo la rivendicazione 1, comprendente inoltre le fasi di aggiornare, rispetto ad ogni fase che viene eseguita, un file con dati relativi allo stato corrente dell'installazione di software; e

prima dell'esecuzione di ogni fase, determinare dal file se tale fase è in modo appropriato la successiva fase consecutiva nella sequenza di fasi.

3. - Procedimento per la prova di un sistema di elaboratore, comprendente le fasi di prevedere una sequenza di fasi comprendente una pluralità di fasi di prova da eseguire in un ordine determinato dalla sequenza di fasi, e

leggere ed eseguire fasi consecutive dalla sequenza di fasi.

4. - Procedimento secondo la rivendicazione 3, comprendente inoltre le fasi di aggiornare, rispetto

ad ogni fase che è eseguita, un file con dati relativi allo stato corrente delle prove; e

prima dell'esecuzione di ogni fase, determinare dal file se tale fase è in modo appropriato la successiva fase consecutiva nella sequenza.

5. - Procedimento per l'installazione di software e la prova di un sistema di elaboratore, comprendente le fasi di prevedere una sequenza di fasi comprendente una pluralità di fasi di installazione di software e di test da eseguire in un ordine determinato dalla sequenza di fasi; e

leggere ed eseguire fasi consecutive dalla sequenza di fasi.

6. - Procedimento secondo la rivendicazione 5, comprendente inoltre le fasi di aggiornare, rispetto ad ogni fase che è eseguita, un file con dati relativi allo stato corrente dell'installazione di software e delle prove; e

prima dell'esecuzione di ogni fase, determinare dal file se tale fase è in modo appropriato la successiva fase consecutiva nella sequenza di fasi.

7. - Procedimento per l'installazione di software e/o per la prova di un sistema di elaboratore, il procedimento comprendendo le fasi di:

prevedere una sequenza di fasi comprendente una

pluralità di fasi di installazione di software e/o di test da eseguire in un ordine determinato dalla sequenza di fasi;

leggere ed eseguire fasi consecutive dalla sequenza di fasi;

aggiornare, rispetto ad ogni fase che viene eseguita, un file di progresso con dati relativi allo stato corrente dell'installazione e/o dei test; e

prima dell'esecuzione di ogni fase, determinare dal file di progresso se tale fase è in modo appropriato la successiva fase consecutiva nella sequenza di fasi.

8. - Procedimento secondo la rivendicazione 1, in cui il file di progresso contiene il numero di linea nella sequenza di fasi dell'ultima fase eseguita e il comando eseguito in tale fase.

9. - Procedimento secondo la rivendicazione 1, comprendente inoltre la fase di, prima dell'esecuzione di ogni fase, leggere l'elenco di directory di tutti i file sull'unità locale dell'elaboratore che esegue la sequenza di fasi nella memoria di detto elaboratore.

10. - Procedimento secondo la rivendicazione 1, comprendente inoltre la fase di, prima dell'esecuzione di ogni fase, disabilitare

l'interruzione di controllo per impedire ad un operatore di interrompere la sequenza di fasi per saltare una fase.

11. - Procedimento secondo la rivendicazione 1, in cui la fase di prevedere una sequenza di fasi comprende le fasi di leggere una pluralità di descrittori di componente da un file leggibile da elaboratore, ogni descrittore di componente descrivendo un rispettivo componente del sistema di elaboratore, leggere una pluralità di fasi da una banca dati, ogni fase essendo associata ad un descrittore di componente e comprendendo un rispettivo numero di sequenza, e sequenziare la pluralità di fasi in un predeterminato ordine secondo i numeri di sequenza per fornire la sequenza di fasi.

12. - Procedimento secondo la rivendicazione 1, comprendente inoltre la fase di scrivere la sequenza di fasi in un supporto di memoria non volatile configurato in modo da accompagnare il sistema di elaboratore durante la fabbricazione.

13. - Procedimento secondo la rivendicazione 1, in cui la sequenza di fasi è atta a prevedere comandi ripetibili per un periodo di tempo definito.

14. - Procedimento secondo la rivendicazione 1, in cui la sequenza di fasi è atta a prevedere

comandi ripetibili per un numero definito di iterazioni.

ing. Mauro MANCHITELLO
N. Iscriz. ALBO 607
(in proprio e per gli altri)

TO 2000A 000409

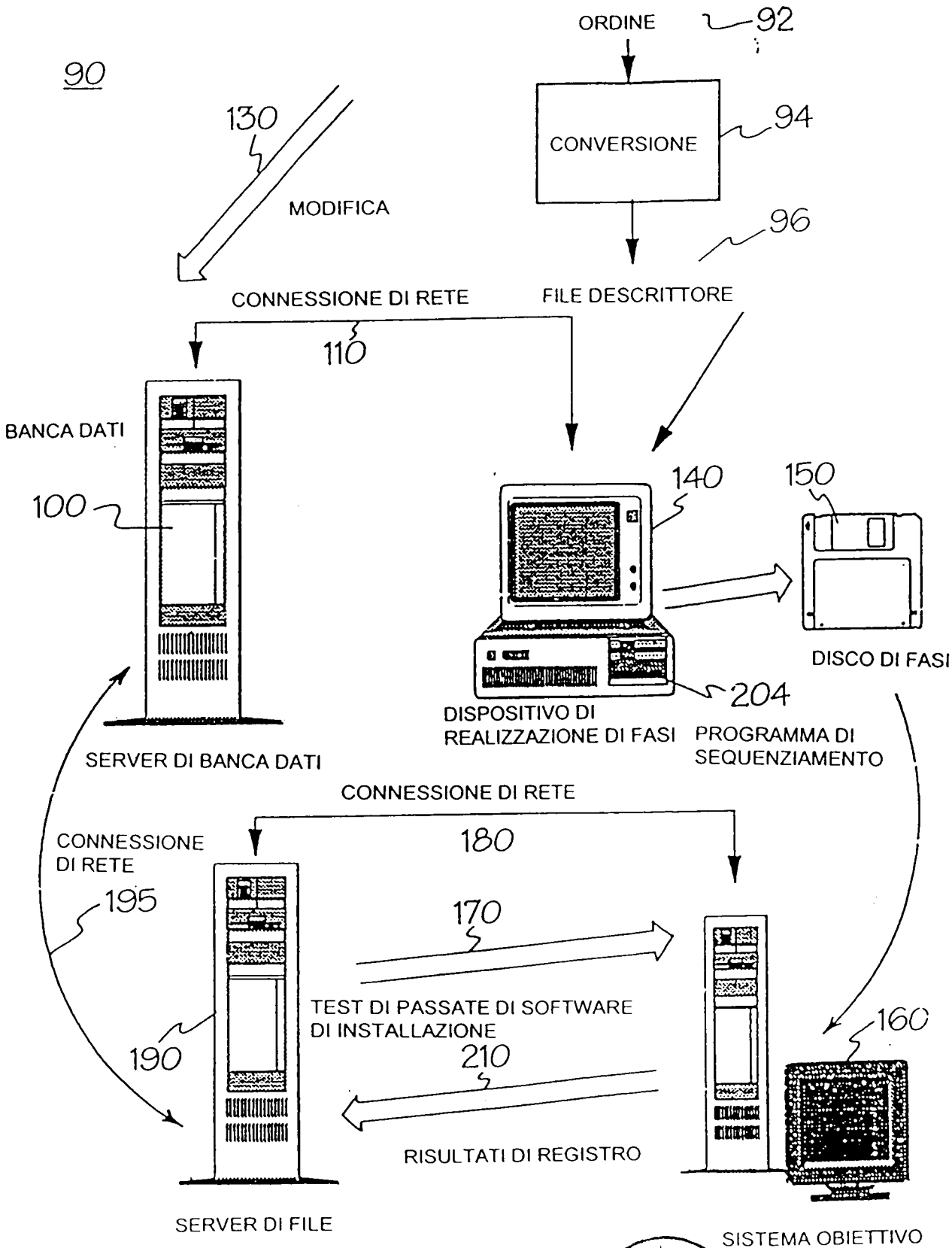


FIG. 1.

TO 2000A 000409

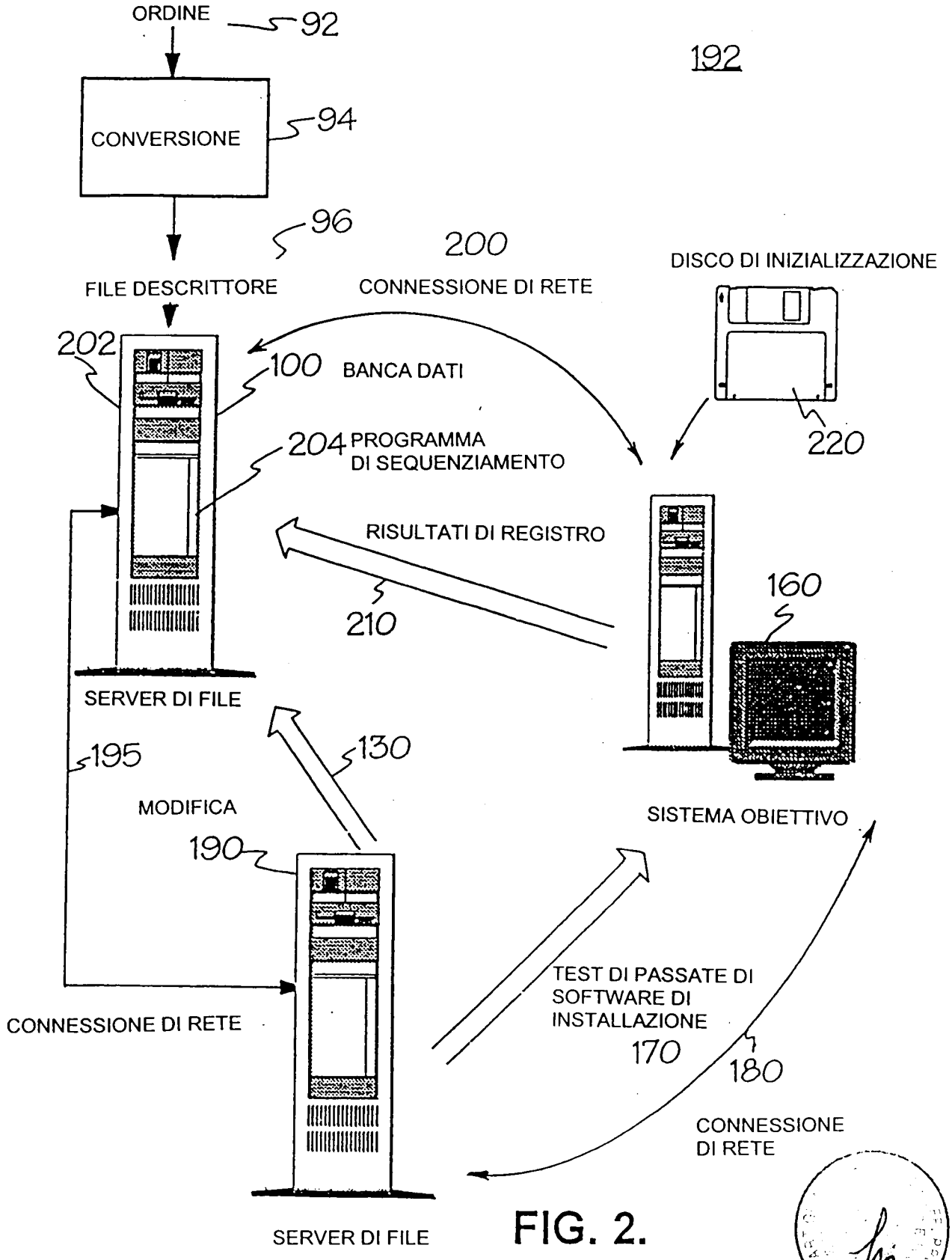


FIG. 2.

Ing. Mauro MARCHITELLI
N. Iscriz. ALBO 507
(in proprio e per gli altri)

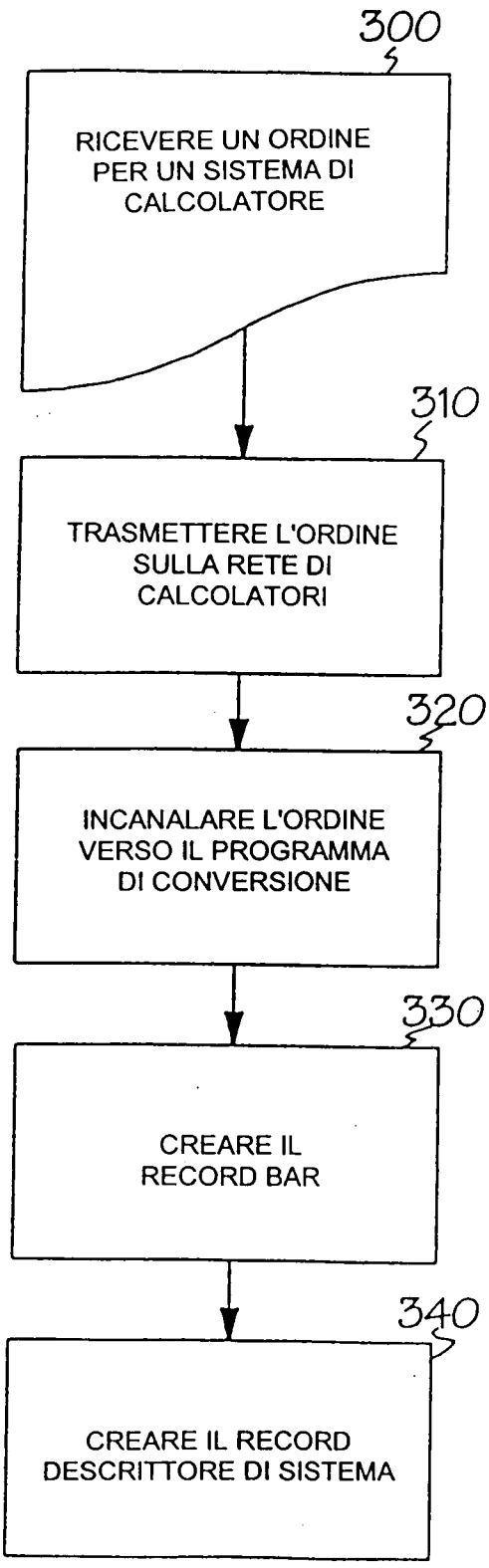


FIG. 3A.

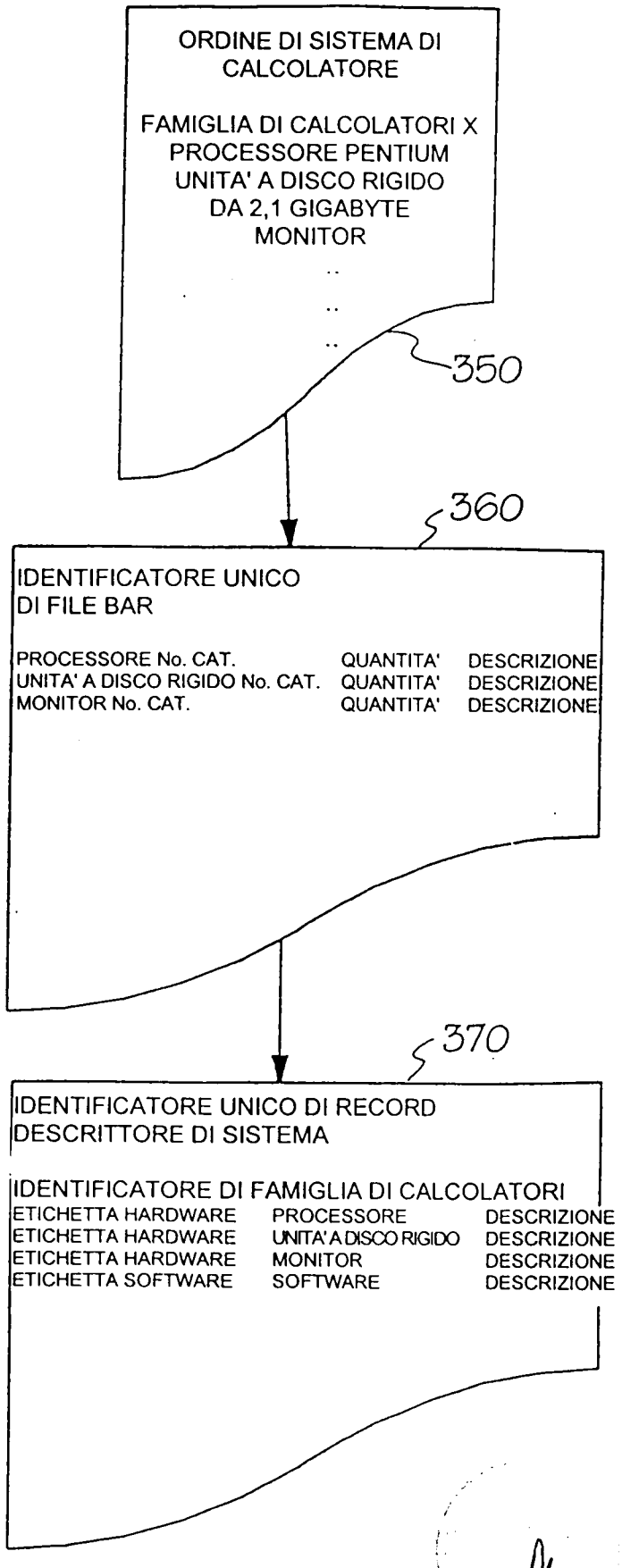


FIG. 3B.

TO 2000A 000409

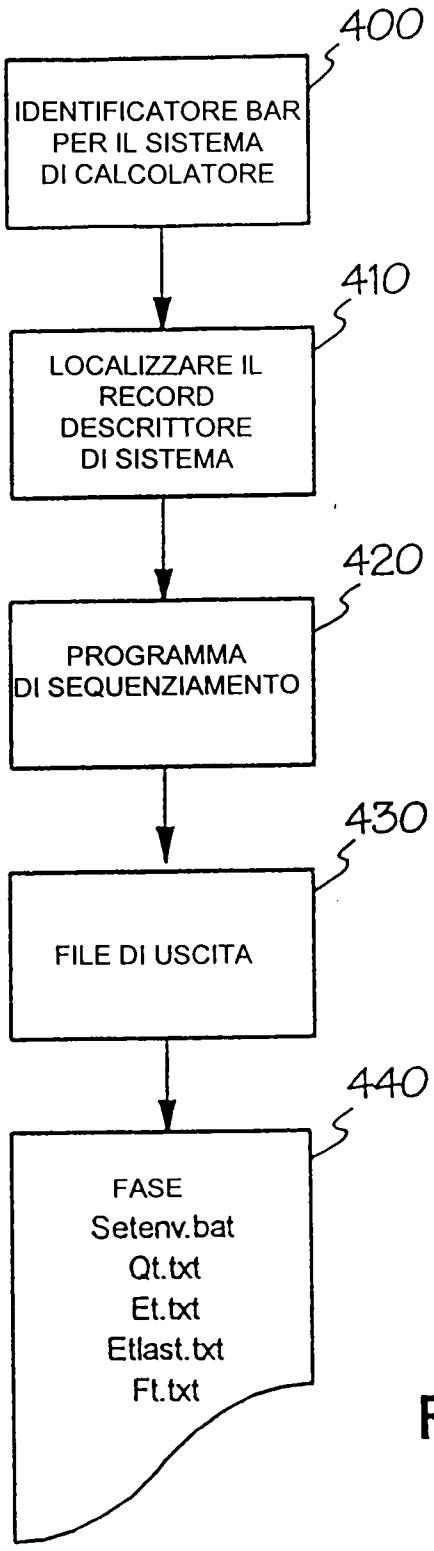


FIG. 4.

TO 2000A 000409

FIG. 5B.

FIG. 5C.

FIG. 5A.

Ing. Mauro MARCHETTI
N. Iscriz. ALBO 507
(in proprio e per gli altri)

TO 2000A 000409

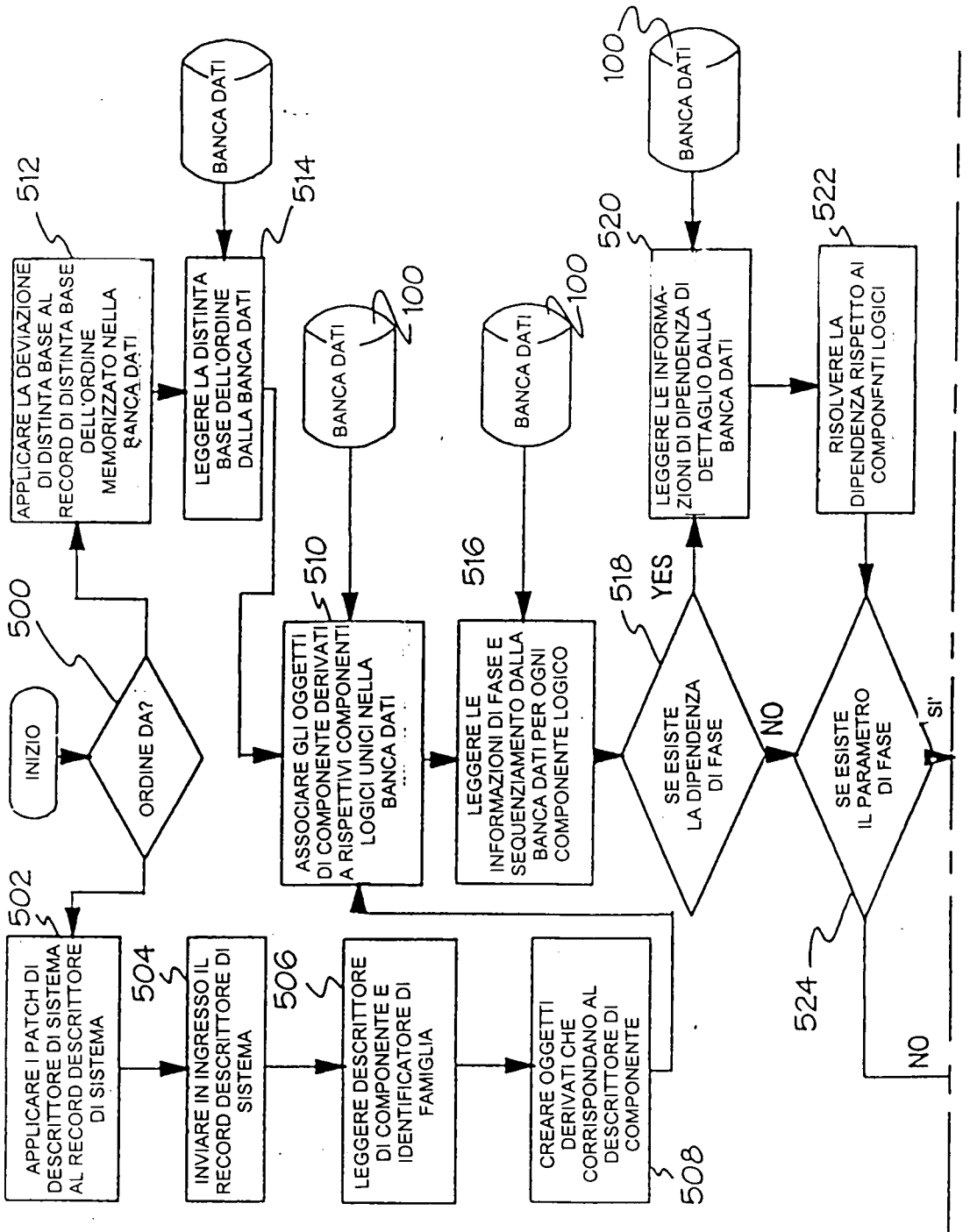


FIG. 5B.

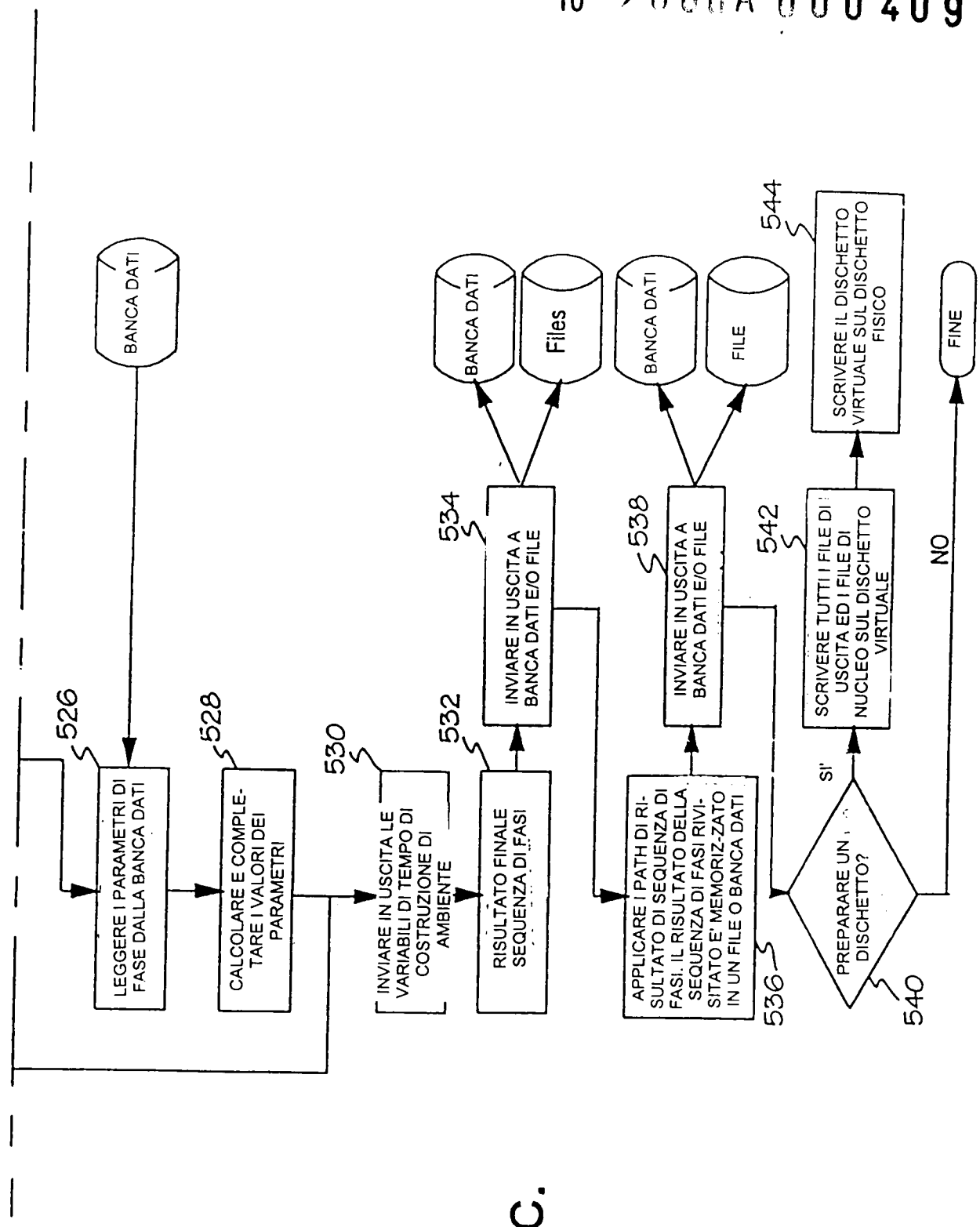


FIG. 5C.

TO 2000A 000409

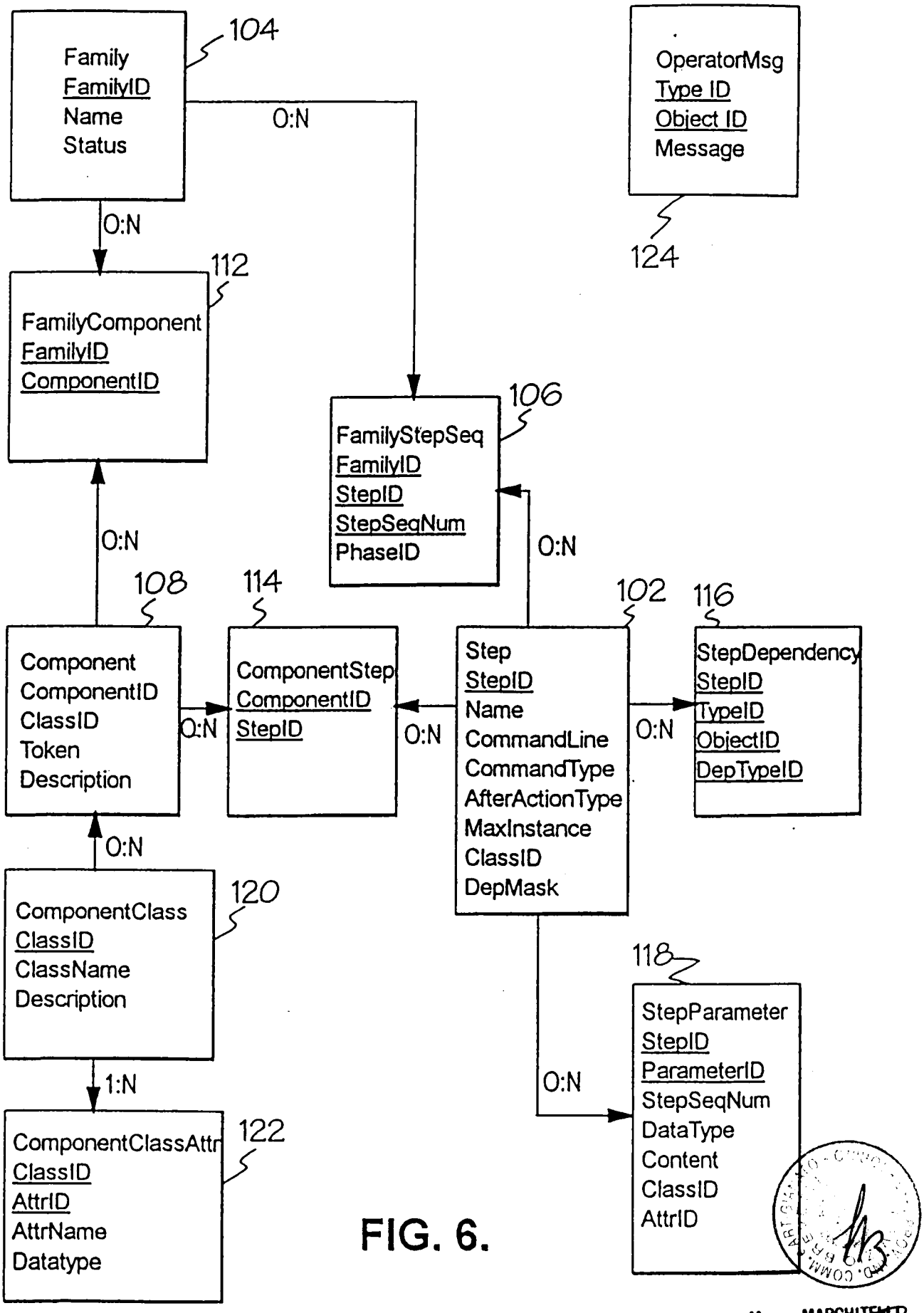


FIG. 6.

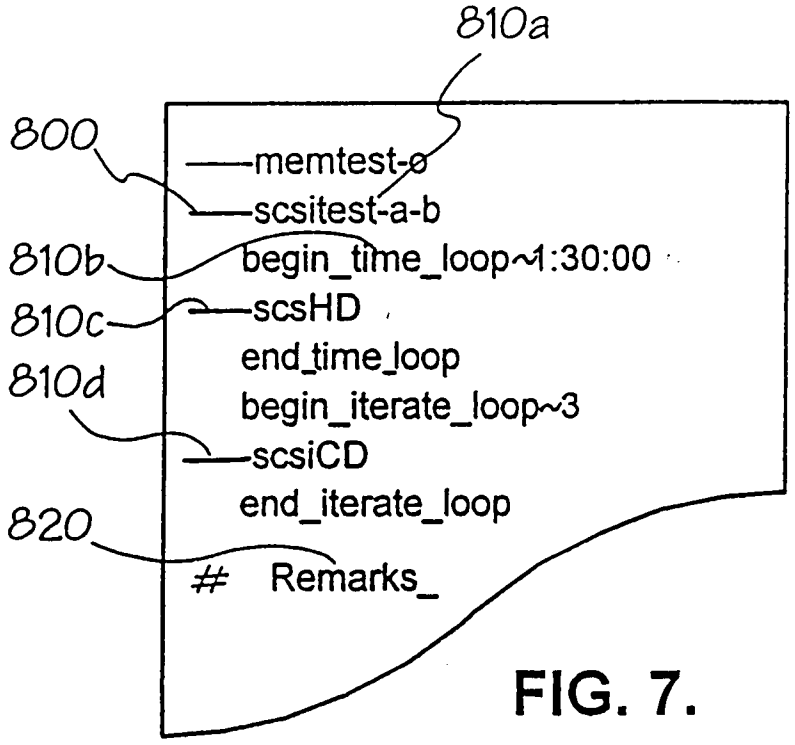


FIG. 7.

RUNSTEP
PAGINA 1

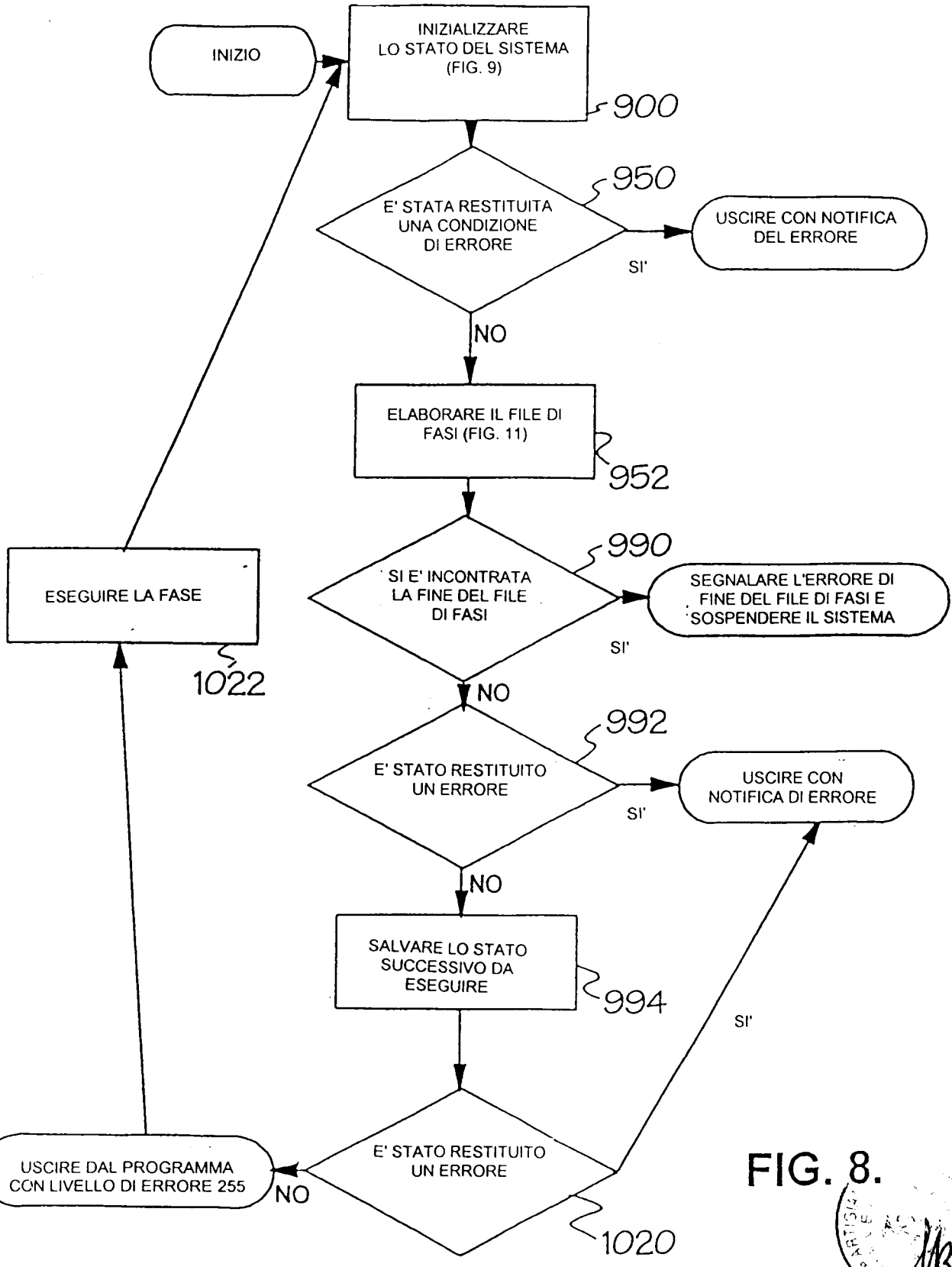


FIG. 8.

TO 2000A 000409

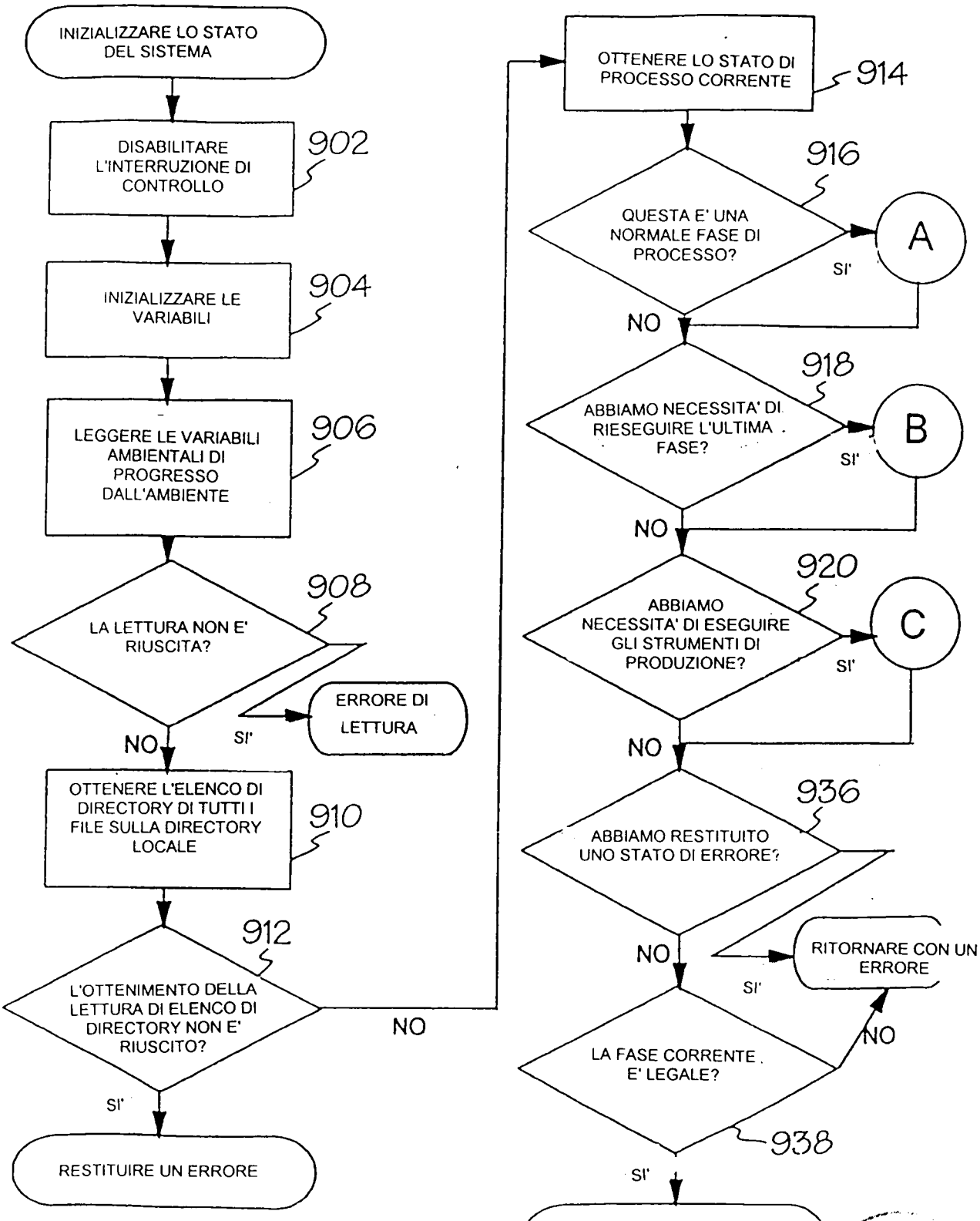
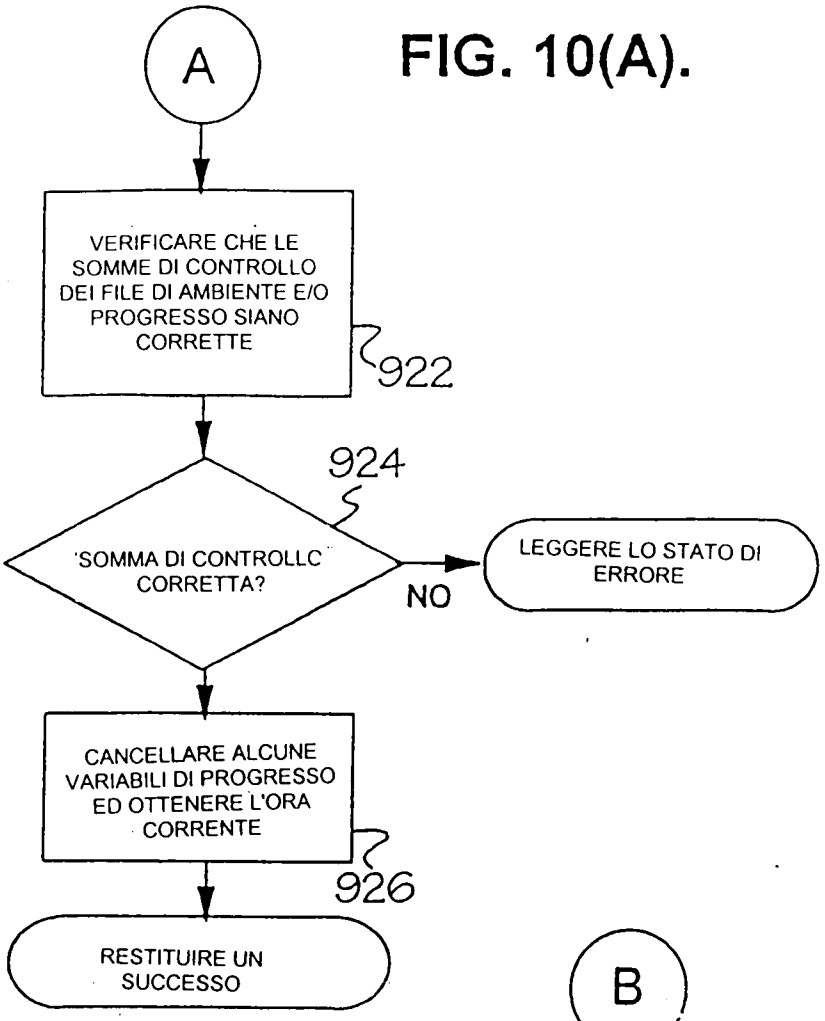


FIG. 9.

FIG. 10(A).



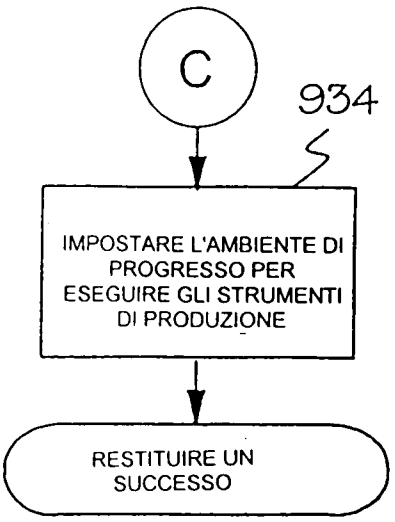


FIG. 10(C).

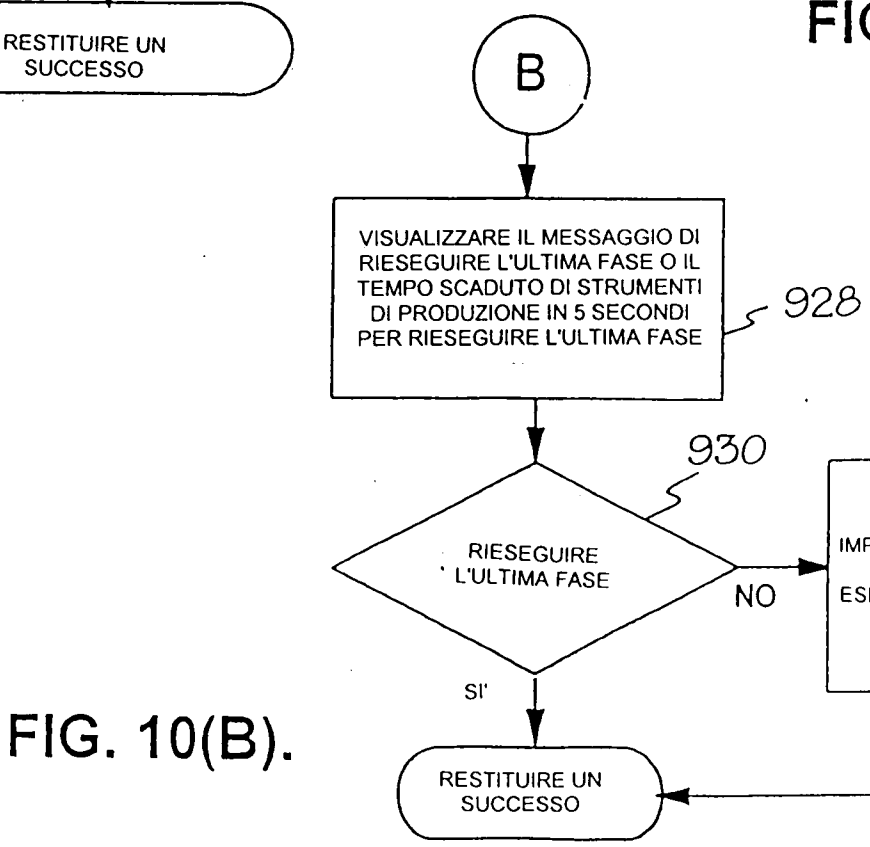


FIG. 10(B).

TO 2000A 000409

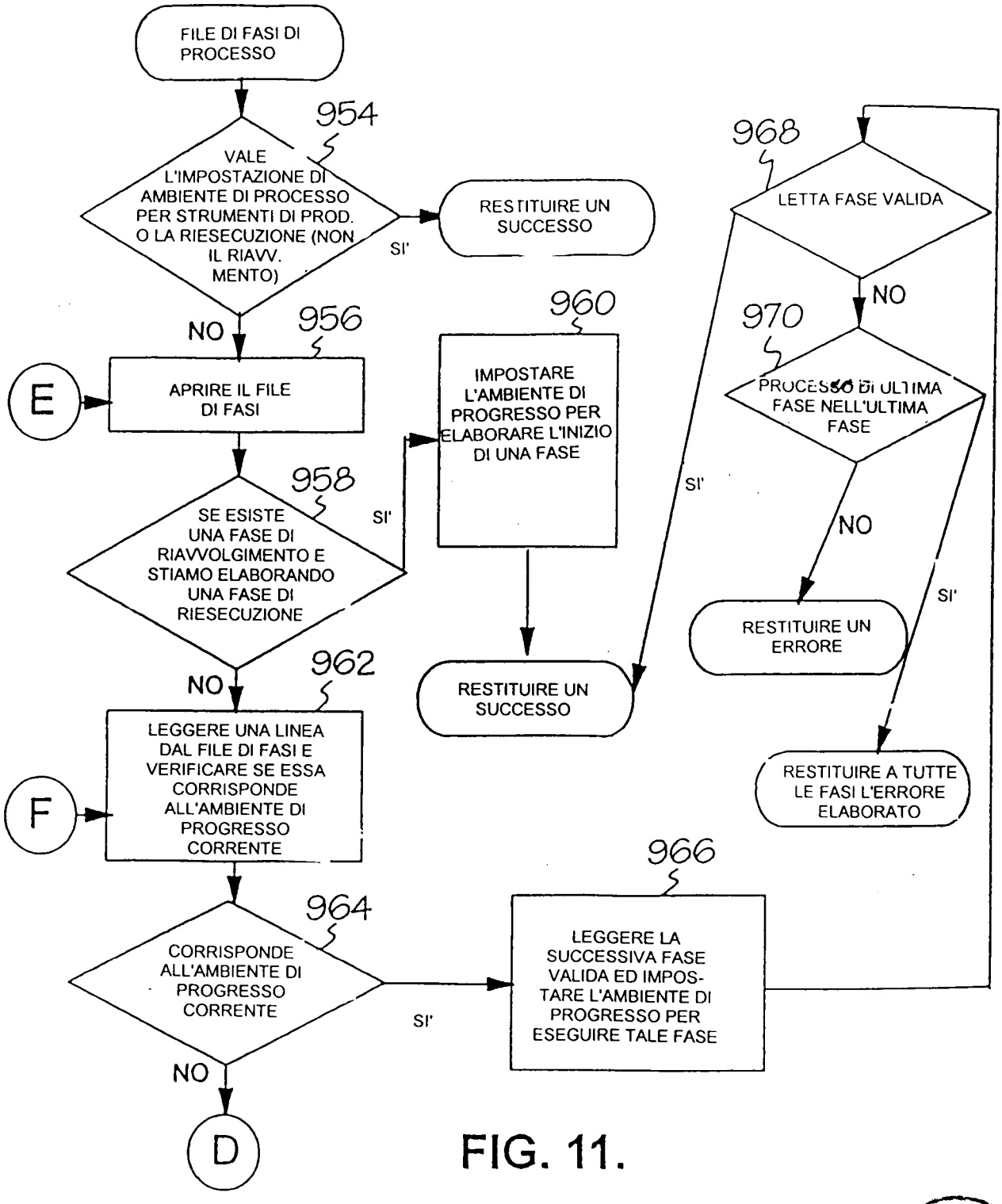


FIG. 11.

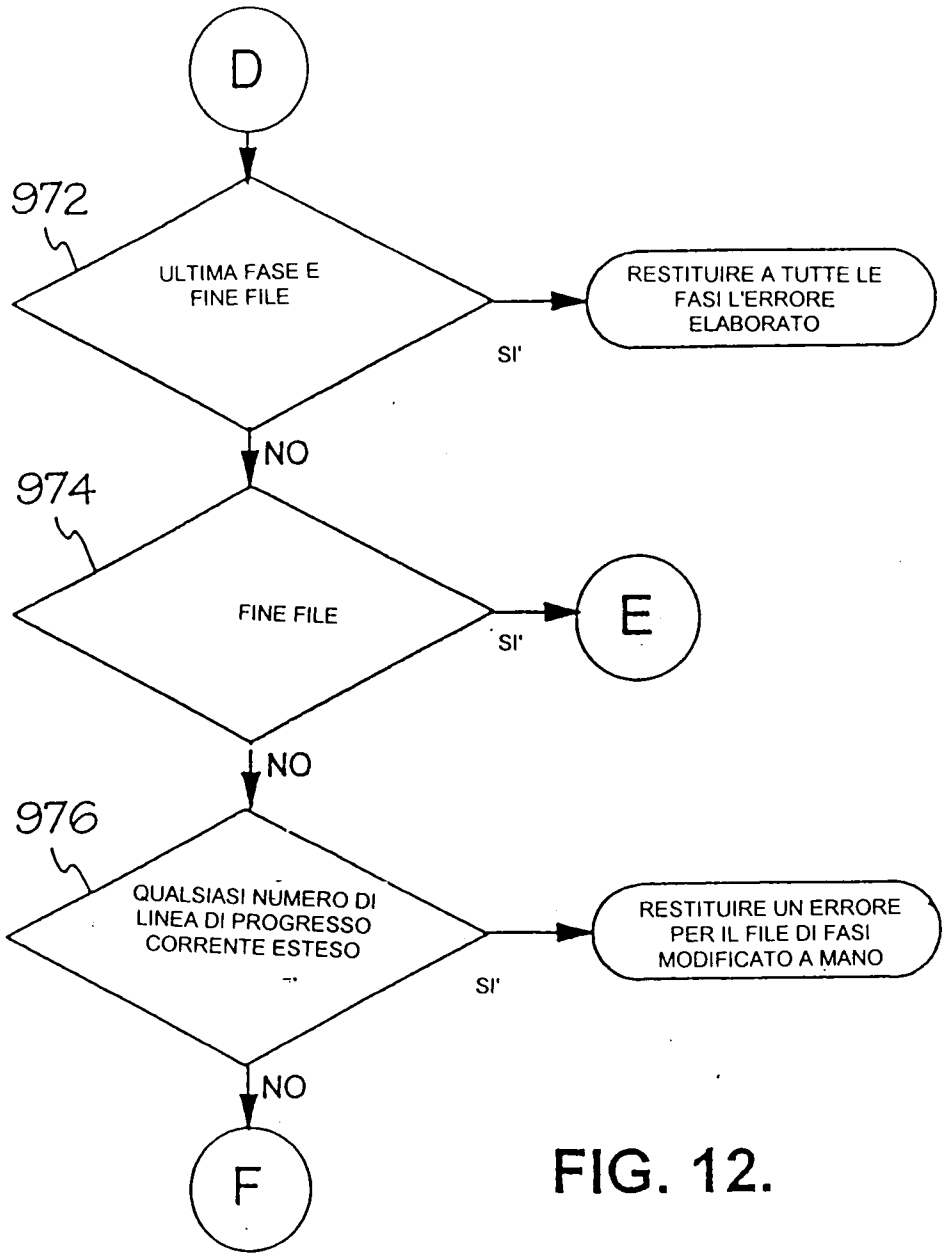


FIG. 12.

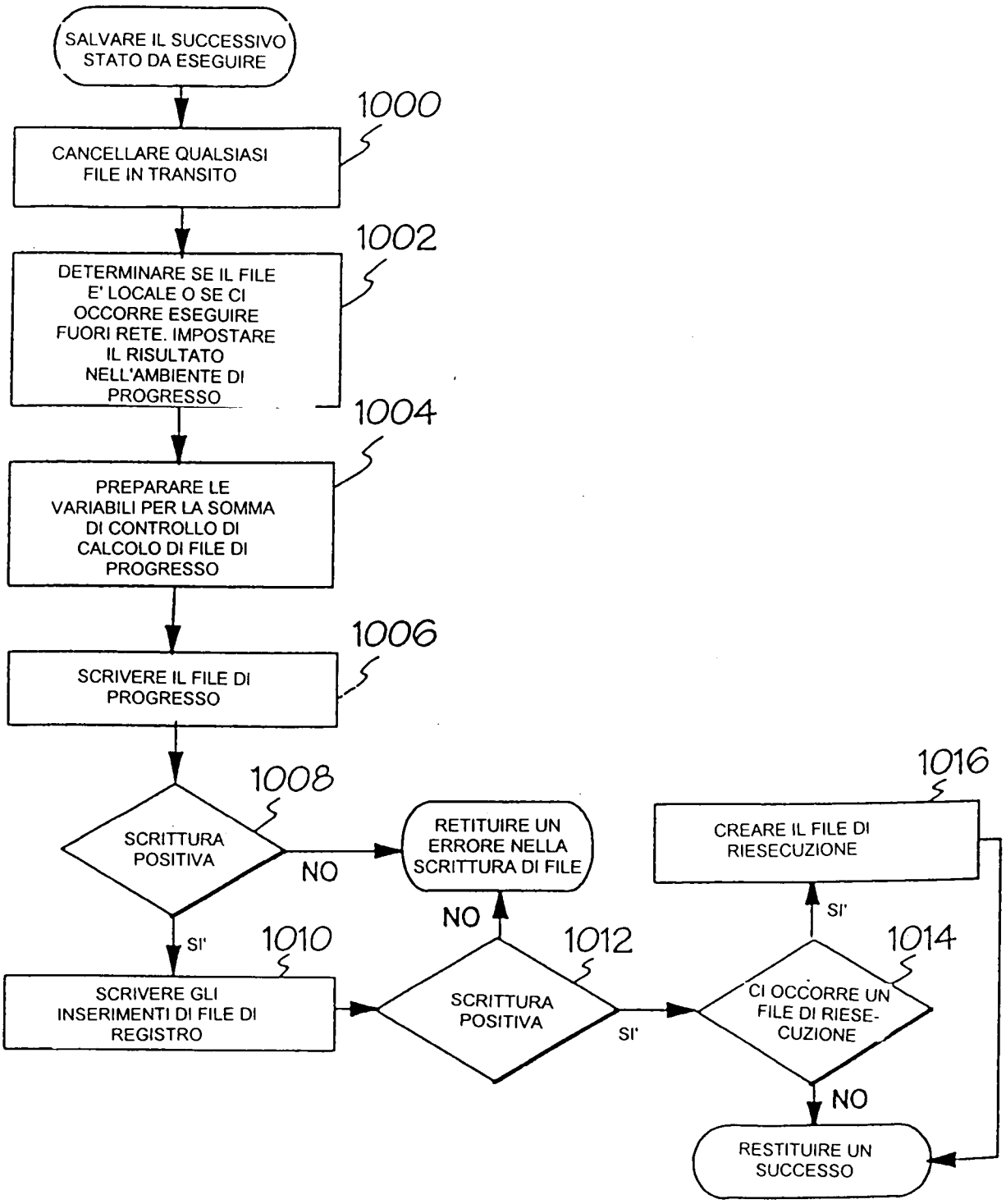


FIG. 13.