(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
17. August 2006 (17.08.2006)

PCT

(10) Internationale Veröffentlichungsnummer
WO 2006/084607 A1

(51) Internationale Patentklassifikation:
G06F 9/44 (2006.01)

(21) Internationales Aktenzeichen: PCT/EP2006/000841

(22) Internationales Anmeldedatum:
31. Januar 2006 (31.01.2006)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(30) Angaben zur Priorität:
05002656.6 9. Februar 2005 (09.02.2005) EP

(71) Anmelder *(für alle Bestimmungsstaaten mit Ausnahme von US)*: **DEUTSCHE POST AG** [DE/DE]; Charles-de-gaulle-str. 20, 53113 Bonn (DE).

(72) Erfinder; und

(75) Erfinder/Anmelder *(nur für US)*: **KRANZ, Jürgen** [DE/DE]; Josef-zimmermann-strasse 15, 50374 Erftstadt (DE). **WESTERWELLER, Jörg** [DE/DE]; Heidebusch 11, 51570 Windeck / Rosbach (DE).

(74) Anwalt: **JOSTARNDT PATENTANWALTS-AG**; Brüsseler Ring 51, 52074 Aachen (DE).

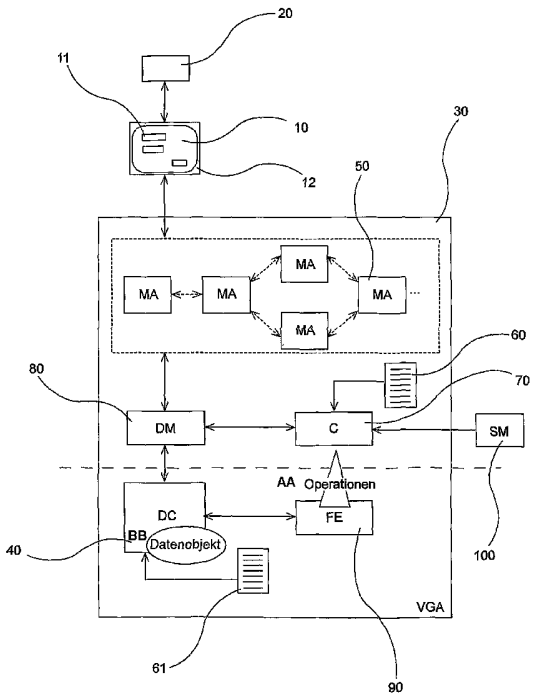
(81) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare nationale Schutzrechtsart)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI,

[Fortsetzung auf der nächsten Seite]

WO 2006/084607 A1

(54) Title: METHOD AND DEVICE FOR THE COMPUTER-ASSISTED GENERATION OF A GRAPHICAL USER INTERFACE

(54) Bezeichnung: VERFAHREN UND VORRICHTUNG ZUM RECHNERGESTÜTZTEN ERZEUGEN EINER GRAPHISCHEN BENUTZEROBERFLÄCHE



AA... OPERATIONS
BB... DATA OBJECT

(57) Abstract: The invention relates to a method and a device for the computer-assisted generation of a graphical user interface (10) comprising at least one control element (11). The user interface is preferably displayed on a display element (12) and is designed to be used interactively by a user (20) together with at least one transaction processing module (30), which permits a transaction to be executed. The method and device are characterised in that the functional unit (90) and the data container (40) of the transaction processing module (30) are separated from the presentation layer, which is in the form of mask working sections (50). In addition, data objects (41) are assigned to the control elements (11) of the user interface (10) via the mask working area (50) and values are assigned to said data objects in the data container (40). When the transaction processing module is activated, a data container configuration file (61) is read, said container containing the configuration for the data and its start initial values that are used by the transaction processing module. A controller configuration file (60) containing the configuration for the operations and the mask flow chart of the transaction processing module is then read.

(57) Zusammenfassung: Die Erfindung betrifft ein Verfahren und eine Vorrichtung zum rechnergestützten Erzeugen einer graphischen Benutzeroberfläche (10) mit wenigstens einem Steuerelement (11). Die Benutzeroberfläche wird vorzugsweise auf einem Anzeigemittel (12) angezeigt und dient zur Interaktion eines Benutzers (20) mit wenigstens einem Vorgangsbearbeitungsmodul (30), mittels dessen eine Transaktion abgewickelt werden kann. Das Verfahren und die Vorrichtung

[Fortsetzung auf der nächsten Seite]

NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare regionale Schutzrechtsart)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), eurasisches (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), europäisches (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Erklärungen gemäß Regel 4.17:

— *hinsichtlich der Berechtigung des Anmelders, ein Patent zu beantragen und zu erhalten (Regel 4.17 Ziffer ii)*

— *Erfindererklärung (Regel 4.17 Ziffer iv)*

Veröffentlicht:

— *mit internationalem Recherchenbericht*

Zur Erklärung der Zweibuchstaben-Codes und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

zeichnen sich dadurch aus, dass die Funktionseinheit (90) und der Datencontainer (40) des Vorgangsbearbeitungsmoduls (30) von der Präsentationsschicht in Form von Maskenarbeitsbereichen (50) voneinander getrennt sind. Ferner sind den Steuerelementen (11) der Benutzeroberfläche (10) über den Maskenarbeitsbereich (50) Datenobjekte (41) zugeordnet, denen im Datencontainer (40) Werte zugeordnet sind. Beim Start des Vorgangsbearbeitungsmoduls wird eine Datencontainer-Konfigurationsdatei (61) eingelesen, welche die Konfiguration für die von dem Vorgangsbeabeitungsmodul verwendeten Daten und deren Start-Werte enthält. Ferner wird eine Controller-Konfigurationsdatei (60) eingelesen, welche die Konfiguration für die Operationen und den Maskenablaufplan des Vorgangsbearbeitungsmoduls beinhaltet.

Verfahren und Vorrichtung zum rechnergestützten Erzeugen einer graphischen Benutzeroberfläche

Beschreibung:

Die Erfindung betrifft ein Verfahren zum rechnergestützten Erzeugen einer graphischen Benutzeroberfläche für ein Vorgangsbearbeitungsmodul, wobei die Benutzeroberfläche wenigstens ein Steuerelement aufweist und das Vorgangsbearbeitungsmodul auf einem Computermittel installiert ist.

Die Erfindung betrifft ferner eine Vorrichtung zur Durchführung des Verfahrens und ein computerlesbares Medium, das Befehle zum Ausführen des Verfahrens auf einem Computermittel aufweist.

Bei Anwendungsprogrammen, welche auf Computermitteln installiert und ausgeführt werden, ist es bekannt, graphische Benutzerschnittstellen einzusetzen, um eine vereinfachte Interaktion eines Benutzers mit dem Anwendungsprogramm zu ermöglichen. Derartige Benutzeroberflächen werden auch als Graphical User Interface (GUI) bezeichnet. Benutzeroberflächen von Anwendungsprogrammen erfüllen zum einen die Funktion, dass Elemente und Informationen auf einem Bildschirm angezeigt werden. Zum anderen müssen sie die Ausführung von Aktionen ermöglichen, welche von dem Benutzer an der Benutzeroberfläche initiiert werden. Wird eine Benutzeroberfläche beispielsweise in einem Retailsystem in Form eines Filial- oder Automatensystems eingesetzt, in welchem verschiedene Transaktionen durchgeführt werden können, wird durch die Ansteuerung von Elementen oder die Eingabe von Werten auf der Benutzeroberfläche die Durchführung der

BESTÄTIGUNGSKOPIE

Transaktionen gesteuert. Üblicherweise enthalten Anwendungsprogramme dazu programmierte GUI-Funktionen, welche den Ablauf und die Ansteuerung der einzelnen GUI-Elemente festlegen und Kernfunktionen, welche die Abwicklung der Transaktionen mit dem Anwendungsprogramm betreffen.

Die meisten Anwendungsprogramme sind so gestaltet, dass sie mit einer definierten Benutzeroberfläche betrieben werden. Dabei ist der Programmcode der GUI-Elemente direkt mit den Steuerelementen der Benutzeroberfläche verbunden. Soll ein Anwendungsprogramm mit einer veränderten oder vollständig anderen Benutzeroberfläche betrieben werden, stellt es daher einen aufwändigen Prozess dar, den Programmcode der Anwendungssoftware zu ändern, um eine geänderte Benutzeroberfläche zu implementieren.

Um jegliche Benutzeroberfläche mit verschiedenen Anwendungsprogrammen nutzen zu können, ist es beispielsweise aus der europäischen Patentanmeldung EP 0 798 634 A1 bekannt, die Funktionen der Benutzeroberfläche von den Kernfunktionen des Anwendungsprogrammes zu trennen. Dazu wird ein objektorientiertes Programmiermodell mit einem so genannten "Look&Feel"-Agenten (L&F-Agent) verwendet. Der Agent agiert als Server für einen oder mehrere logische Objekt-Clients. Ein derartiger Agent steuert die Darstellung und das Verhalten der Benutzeroberfläche, während ein Logikobjekt die Funktionen des Anwendungsprogrammes durchführt. Dem L&F-Agenten liegen dabei keine Informationen über die Funktionen der Kernlogik vor und die Kernlogik weiß nicht, wie die Benutzeroberfläche aussieht oder sich verhält.

Ferner beschreibt die internationale Patentanmeldung WO 2004/107200 A1 ein System und ein Verfahren zur Steuerung

von Elementen einer Benutzeroberfläche mit Daten. Ein Computermittel mit einem Anwendungsprogramm ist dabei in zwei Bereiche aufgeteilt, wodurch der Logikbereich von dem Bereich der Benutzeroberfläche getrennt ausgebildet ist. Eine Verbindungskomponente beschreibt die Beziehung zwischen den Elementen der Benutzeroberfläche und deren Eigenschaften mit Datenwerten aus einer Datenquelle.

Um Systeme bereitzustellen, welche möglichst einfach erweiterbar oder veränderbar sind, ist es ferner bekannt, derartige Systeme modular auszubilden. Bei den Modulen handelt es sich um standardisierte Komponenten, welche von mehreren anderen Komponenten verwendet werden können. Vorzugsweise können neue Komponenten nach dem "Plug&Play"-Prinzip angeschlossen werden, ohne dass das gesamte System angepasst werden muss.

Um neue Vorgänge in bestehende Retailsysteme integrieren zu können, so dass diese an graphischen Benutzeroberflächen darstellbar und mit einem Anwendungsprogramm durchführbar sind, müssen neue Komponenten in das bestehende System integriert werden. Bei Retailsystemen im Bereich des Versicherungs-, Finanz- und Logistikwesens müssen beispielsweise neue Vorgänge wie die Eröffnung eines Bankkontos, der Abschluss eines Versicherungsvertrages, der Verkauf eines Tickets oder die Einlieferung eines Versandpaketes abgebildet werden. Dies erfordert insbesondere die Erzeugung einer entsprechenden Benutzeroberfläche zur Interaktion eines Benutzers mit dem Anwendungsprogramm, wobei zu beachten ist, dass in einem Retailsystem unterschiedliche Vorrichtungen zur Darstellung der Benutzeroberfläche verwendet werden können. Dazu zählen beispielsweise Computermonitore,

Automatendisplays, Displays von Mobiltelefonen oder Web-Sites.

Aufgabe der Erfindung ist es, ein Verfahren zum rechnergestützten Erzeugen einer graphischen Benutzeroberfläche bereitzustellen, das die Verwendung variabler Benutzeroberflächen für ein Anwendungsprogramm ermöglicht. Dabei soll berücksichtigt werden, dass das Verfahren auf einfache Weise die modulartige Erweiterung eines Gesamtsystems ermöglicht, in welchem mehrere Vorgänge eines Retailsystems abgebildet werden.

Aufgabe der Erfindung ist es ferner, eine Vorrichtung zur Durchführung des Verfahrens bereitzustellen.

Erfindungsgemäß wird diese Aufgabe durch ein Verfahren mit den Merkmalen des unabhängigen Anspruches 1 gelöst. Vorteilhafte Weiterbildungen des Verfahrens ergeben sich aus den Unteransprüchen 2-8. Die Aufgabe wird ferner durch eine Vorrichtung nach Anspruch 9 gelöst. Vorteilhafte Ausführungsformen der Vorrichtung ergeben sich aus den Unteransprüchen 10-12. Anspruch 13 gibt ein computerlesbares Medium zur Durchführung des Verfahrens an.

Bei dem erfindungsgemäßen Verfahren zum rechnergestützten Erzeugen einer graphischen Benutzeroberfläche weist die Benutzeroberfläche mehrere Steuerelemente auf, welche abgearbeitet und/oder befüllt werden. Beispielsweise werden in einem Assistent-basierten System Steuerelemente mit Werten befüllt oder Schalt- oder Funktionsflächen betätigt. Die Steuerelemente der Benutzeroberfläche können dabei zur Interaktion eines Benutzers mit einem Vorgangsbearbeitungsmodul dienen, wobei das Vorgangsbearbeitungsmodul auf einem

Computermittel installiert ist, das in Verbindung mit einem Anzeigemittel steht. In diesem Fall werden die Steuerelemente über das Anzeigemittel von einem Nutzer befüllt oder betätigt. Die Befüllung von Steuerelementen kann dabei neben einer manuellen Eingabe auch über das Einlesen von maschinenlesbaren Informationen wie beispielsweise Barcodes erfolgen. Die Steuerelemente bzw. die damit verbundenen Datenobjekte können ferner durch den Inhalt einer Datenquelle befüllt werden, deren Werte den Steuerelementen zuzuordnen sind. Bei der Datenquelle handelt es sich beispielsweise um eine Datei, welche unabhängig von systemeigenen Vorgangsbearbeitungsmodulen von einer Vorrichtung erzeugt wurde. Dies kann beispielsweise bei einem Automaten der Fall sein, auf welchem ein Vorgangsbearbeitungsmodul und der zugehörige Maskenarbeitsbereich nicht lauffähig sind, dessen erfasste Daten jedoch einheitlich im System der Vorgangsbearbeitungsmodule verarbeitet werden sollen.

Das erfindungsgemäße Verfahren sieht das Zuordnen von wenigstens einem Maskenarbeitsbereich mit wenigstens einem Steuerelement zu einem Vorgangsbearbeitungsmodul vor, wobei jedem Steuerelement ein Datenobjekt zugeordnet wird. Das Vorgangsbearbeitungsmodul wird durch ein Startmodul gestartet, wobei beim Starten wenigstens ein Controller, eine Funktionseinheit und ein Datencontainer des Vorgangsbearbeitungsmoduls initialisiert werden. Dabei wird eine Controller-Konfigurationsdatei zur Initialisierung des Controllers eingelesen, wobei die Controller-Konfigurationsdatei die Konfiguration der Darstellung des Maskenarbeitsbereiches des betreffenden Vorgangsbearbeitungsmoduls beinhaltet. Ferner wird eine Datencontainer-Konfigurationsdatei zur Initialisierung des Datencontainers eingelesen, wobei die Datencontainer-Konfigurationsdatei die Konfiguration der

Datenobjekte des Datencontainers des betreffenden Vorgangsbearbeitungsmoduls und die Konfiguration der Operationen für die Kommunikation zwischen dem Maskenarbeitsbereich, der Funktionseinheit und dem Datencontainer beinhaltet. Das Verfahren sieht ferner das Initialisieren einer Darstellungskomponente durch den Controller und das Ermitteln und Initialisieren des Maskenarbeitsbereiches des Vorgangsbearbeitungsmoduls durch den Controller vor. Die Steuerelemente des Maskenarbeitsbereiches werden initialisiert und können durch einen Nutzer oder eine eingespeiste Datenquelle befüllt und/oder betätigt werden. Die Controller-Konfigurationsdatei und/oder die Datencontainer-Konfigurationsdatei werden vorzugsweise im XML-Format (Extensible Markup Language) bereitgestellt.

In einem besonders bevorzugten Ausführungsbeispiel der Erfindung werden die Steuerelemente als graphische Benutzeroberfläche auf einem Anzeigemittel dargestellt und dienen zur Interaktion eines Benutzers mit einem Vorgangsbearbeitungsmodul. Die Steuerelemente werden befüllt und/oder abgearbeitet, wobei der Maskenarbeitsbereich über die Darstellungskomponente auf die Datenobjekte des Datencontainers zugreift.

In einem weiteren Ausführungsbeispiel der Erfindung werden die Steuerelemente nicht auf einem Anzeigemittel angezeigt, sondern eine Datenquelle wird über eine Schnittstelle eingelesen, wobei der Inhalt der Datenquelle den Datenobjekten des Datencontainers eines Vorgangsbearbeitungsmodules zugeordnet werden kann. Die Steuerelemente des Maskenarbeitsbereiches werden in diesem Fall befüllt, indem die Steuerelemente und damit die Datenobjekte des Datencontainers automatisch mit Werten der Datenquelle gefüllt werden. Auch die Abarbeitung von Steuerelementen wie Schalt-

oder Funktionsflächen erfolgt automatisch, so dass der Maskenarbeitsbereich des Vorgangsbearbeitungsmoduls abgearbeitet werden kann, als ob er auf einem Anzeigemittel von einem Nutzer durchlaufen würde.

Die Funktionseinheit hat keine Kenntnis von den Steuerelementen und hat folglich auch keine unmittelbare Kontrolle über die Objekte der Benutzeroberfläche. Gleichermaßen wirken Änderungen von Feldinhalten in den Steuerelementen nicht unmittelbar auf die Funktionseinheit. Die Funktionseinheit ist vorzugsweise frei von einem bestimmten Status, da sich der Status eines Vorgangsbearbeitungsmoduls in den konkreten Werten der Datenobjekte im Datencontainer widerspiegelt. Die Ausführung der Funktionseinheit ohne Status hat insbesondere den Vorteil, dass die Funktionseinheit dadurch flexibel im Frontend oder im Backend eines Gesamtsystems angeordnet werden kann.

Bei einer Änderung eines Wertes innerhalb der Funktionseinheit, wird das zugeordnete Datenobjekt, das den Wert enthält, über die Darstellungskomponente an den Maskenarbeitsbereich und alle Steuerelemente weitergeleitet, welche den Wert darstellen. Dabei erhält die Funktionseinheit die Kontrolle, was es ihr ermöglicht, Berechnungen, Dateizugriffe, etc. durchzuführen. Insbesondere kann die Funktionseinheit ein laufendes Vorgangsbearbeitungsmodul pausieren lassen und ein neues Vorgangsbearbeitungsmodul starten. Bei einer Änderung eines Wertes auf der Benutzeroberfläche leitet das entsprechende Steuerelement den Wert an die Funktionseinheit und den Datencontainer weiter, wobei der Wert zu dem Datenobjekt gespeichert wird, welches dem Steuerelement zugeordnet ist. Die Funktionseinheit manipuliert die Datenobjekte und das Steuerelement stellt den konkreten aktuellen Inhalt und

Zustand des Datenobjektes dar, nachdem die Kontrolle von der Funktionseinheit wieder an den Maskenarbeitsbereich übergegangen ist. Diese Verfahrensschritte werden sowohl bei einer Abarbeitung durch einen Benutzer auf einem Anzeigemittel als auch bei der automatischen Abarbeitung einer über eine Schnittstelle eingespeisten Datenquelle durchgeführt.

In einem besonders bevorzugten Ausführungsbeispiel der Erfindung werden auf dem Computermittel mehrere Vorgangsbearbeitungsmodule installiert und ein erstes Vorgangsbearbeitungsmodul ruft ein zweites Vorgangsbearbeitungsmodul auf, indem das gemeinsame Startmodul das zweite Vorgangsbearbeitungsmodul startet und der Maskenarbeitsbereich des zweiten Vorgangsbearbeitungsmoduls initialisiert wird, wobei der zweite Maskenarbeitsbereich bei der Initialisierung und Abarbeitung der Steuerelemente über seine Darstellungskomponente auf Inhalte seines Datencontainers zugreift.

Die Erfindung umfasst ferner eine Vorrichtung zum rechnergestützten Erzeugen einer graphischen Benutzeroberfläche mit mehreren Steuerelementen. Die Steuerelemente können auf einem Anzeigemittel dargestellt werden und zur Interaktion eines Benutzers mit einem Vorgangsbearbeitungsmodul dienen, wobei das Vorgangsbearbeitungsmodul auf einem Computermittel installiert ist, das in Verbindung mit dem Anzeigemittel steht. Das Vorgangsbearbeitungsmodul umfasst dabei wenigstens einen Maskenarbeitsbereich, einen Controller, einen Datencontainer und eine Funktionseinheit. Die Vorrichtung weist ferner ein Startmodul zum Starten des Vorgangsbearbeitungsmoduls auf, wobei das Startmodul Mittel zum Initialisieren wenigstens des Controllers, der Funktionseinheit und des Datencontainers aufweist. Der Controller umfasst wenigstens Mittel zum Einlesen einer Controller-Konfigurationsdatei,

Mittel zum Initialisieren einer Darstellungskomponente sowie Mittel zum Ermitteln und Initialisieren des Maskenarbeitsbereiches des Vorgangsbearbeitungsmoduls. Der Maskenarbeitsbereich umfasst wenigstens Mittel zum Initialisieren der Steuerelemente des Maskenarbeitsbereiches auf der graphischen Benutzeroberfläche und Mittel zum Zugreifen auf Datenobjekte des Datencontainers über die Darstellungskomponente.

In einem weiteren Ausführungsbeispiel der Erfindung umfasst die Vorrichtung eine Schnittstelle zum Einlesen einer Datenquelle und Mittel zum Zuordnen des Inhaltes der Datenquelle zu Datenobjekten des Datencontainers des Vorgangsbearbeitungsmoduls. Die Vorrichtung umfasst ferner Mittel zur Befüllung von Datenobjekten, welche Steuerelementen zugeordnet sind, mit Werten der Datenquelle.

In einem besonders bevorzugten Ausführungsbeispiel der Vorrichtung sind auf dem Computermittel mehrere Vorgangsbearbeitungsmodule installiert, welche jeweils wenigstens eine Controller-Konfigurationsdatei, eine Datencontainer-Konfigurationsdatei, einen Maskenarbeitsbereich, eine Funktionseinheit und einen Datencontainer aufweisen, wobei ein Vorgangsbearbeitungsmodul jeweils einen fachlichen Vorgang eines Retailsystems abbildet.

Das erfindungsgemäße Verfahren zum rechnergestützten Erzeugen einer graphischen Benutzeroberfläche auf einem Anzeigemittel hat zusammen mit der zugehörigen Vorrichtung den Vorteil, dass die Funktionseinheit und der Datencontainer des Vorgangsbearbeitungsmoduls getrennt von der Präsentationsschicht des Moduls in Form des Maskenarbeitsbereiches ausgebildet ist. Den Steuerelementen des Maskenarbeitsbereiches werden Datenobjekte zugeordnet, die Werte der Datenobjekte befinden

sich jedoch in einem Datencontainer, welcher keine Informationen über die zugehörigen Steuerelemente beinhaltet. Auch die Funktionseinheit des Moduls beinhaltet keine Informationen über die Steuerelemente des Vorgangsbearbeitungsmoduls.

Gemäß der Aufgabe der Erfindung eignet sich das Verfahren insbesondere für die modulartige Erweiterung eines Gesamtsystems, in welchem mehrere Vorgänge eines Retailsystems abgebildet werden. Werden mehrere Vorgangsbearbeitungsmodule verwendet, können diese die gleichen Steuerelemente aufweisen, wobei die vorgangsspezifische Verwendung der Elemente abhängig vom jeweiligen Vorgangsbearbeitungsmodul ist. Dies hat den Vorteil, dass die Verwendung der Steuerelemente verändert werden kann, ohne dass die Benutzeroberfläche umprogrammiert werden muss.

Die Verwendung der Steuerelemente ist ferner nicht in der Funktionseinheit oder dem Datencontainer enthalten, sondern wird über eine jeweilige Konfigurationsdatei bei jedem Start eines Vorgangsbearbeitungsmoduls eingelesen. Die Verwendung der Steuerelemente und die Steuerung des Maskenflusses können somit durch eine einfache Veränderung der Konfigurationsdateien geändert werden, ohne dass die Funktionseinheit oder der Datencontainer umprogrammiert werden müssen. Dies hat den Vorteil, dass die Logik zur Abbildung eines Prozesses nicht verändert werden muss, wenn sich die Benutzeroberfläche ändert, so dass beispielsweise ein Programmierer für die Benutzeroberfläche keine Kenntnis über die Logik der Programmierung des Vorgangs haben muss. Eine Benutzeroberfläche kann daher in Form von Steuerelementen und zugeordneten Datenobjekten über ein einfaches Anwendungsprogramm

mit verschiedenen Tools erstellt werden, wobei das Programm die entsprechende XML-Datei erzeugt.

Die Verwendung einer Darstellungskomponente dient insbesondere zur Bildung einer Abstraktionsebene zwischen den GUI-Steuerelementen der Benutzeroberfläche und einem Programmcode. Erfolgt auf der Benutzeroberfläche beispielsweise die Eingabe eines Wertes, wird dies zwar registriert, der Vorgang ist jedoch nicht direkt mit einem Programmcode verbunden. Auch bei der Betätigung eines Schaltelementes auf der Benutzeroberfläche werden damit verbundene Operationen aktiviert, die jedoch keine direkte Verbindung zu einem Programmcode haben. Dies hat den Vorteil, dass dadurch die GUI-Steuerelemente austauschbar sind, ohne dass die Konfigurationsdateien eines Vorgangsbearbeitungsmoduls geändert werden müssen.

Das erfindungsgemäße Verfahren und die zugehörige Vorrichtung bringen ferner den Vorteil mit sich, dass Vorgaben zur Prüfung von Inhalten eines GUI-Steuerelementes verändert werden können, ohne dass die Funktionseinheit oder das Steuerelement eines Vorgangsbearbeitungsmoduls verändert werden müssen. Soll beispielsweise die Maximallänge eines einzugebenden Wertes in einem GUI-Steuerelement reduziert werden, muss lediglich die Datencontainer-Konfigurationsdatei verändert werden, in welcher diese Vorgaben festgelegt werden. Herkömmliche Verfahren verwenden beispielsweise eine direkte Bindung (binding) zwischen einem GUI-Element und einem Datenobjekt, was dazu führt, dass die Änderung von Wertvorgaben eine Veränderung der Bindung erfordert. Auch die Änderung eines Steuerelementes beispielsweise von einem Textfeld in eine Schaltfläche (Button) erfordert herkömmlicherweise die Anpassung der Bindung. Dies ist jedoch mit

dem erfindungsgemäßen Aufbau ebenfalls durch eine Veränderung der Datencontainer-Konfigurationsdatei eines Vorgangsbearbeitungsmoduls möglich, ohne dass die Funktionseinheit oder ein Maskenarbeitsbereich verändert werden müssen.

Die durch die Erfindung realisierte Trennung von Steuerelementen einer Benutzeroberfläche von einem Programmcode hat ferner den Vorteil, dass der Inhalt einer extern erzeugten Datenquelle eingespeist und mit einem Vorgangsbearbeitungsmodul verarbeitet werden kann. Die Datenquelle kann über eine Schnittstelle eingespeist werden und die Datenobjekte eines Maskenarbeitsbereiches eines Vorgangsbearbeitungsmoduls können befüllt werden, als ob die Werte der Datenquelle von einem Vorgangsbearbeitungsmodul erfasst worden wären. Dabei ist keine reale Befüllung von Feldern auf einer Benutzeroberfläche erforderlich, wie es bei herkömmlichen Verfahren der Fall ist. Da direkt hinter den Steuerelementen kein Programmcode steht, welcher eingegebene Werte oder Funktionsbefehle verarbeitet, sondern die Zuordnung der Steuerelemente zu Datenobjekten und die Verarbeitung der Daten von den GUI-Controls getrennt ist, können die Steuerelemente eines Maskenarbeitsbereiches durchlaufen und die zugeordneten Datenobjekte befüllt werden, ohne dass eine reale Manipulation von Steuerelementen auf einer Benutzeroberfläche erforderlich ist.

Weitere Vorteile, Besonderheiten und zweckmäßige Weiterbildungen der Erfindung ergeben sich aus den Unteransprüchen und der nachfolgenden Darstellung bevorzugter Ausführungsbeispiele anhand der Abbildungen.

Von den Abbildungen zeigt

Fig. 1 eine schematische Darstellung eines besonders bevorzugten Ausführungsbeispiels der erfindungsgemäßen Vorrichtung; und

Fig. 2 eine schematische Darstellung eines Computersystems mit mehreren Vorgangsbearbeitungsmodulen.

In der Fig. 1 ist schematisch ein besonders bevorzugtes Ausführungsbeispiel der erfindungsgemäßen Vorrichtung und der zugehörigen erfindungsgemäßen Verfahrensabläufe dargestellt. Weitere vorteilhafte Ausführungsformen der Erfindung sollen anhand dieser Darstellung erläutert werden.

In der Fig. 1 ist ein Vorgangsbearbeitungsmodul 30 dargestellt, das im Folgenden auch mit VGA bezeichnet wird. Das Vorgangsbearbeitungsmodul bildet elektronisch einen Vorgang oder Geschäftsprozess ab, welcher von dem Vorgangsbearbeitungsmodul abgewickelt wird. Bei den Vorgängen handelt es sich beispielsweise um Transaktionen innerhalb eines Retailsystems, welche im Bereich des Finanz-, Versicherungs-, und Logistikwesens eingesetzt werden. Unter Transaktionen können neben Vorgängen mit einem eindeutigen Anfang und einem eindeutigen Ende, die entweder vollständig oder gar nicht ausgeführt werden, auch statusunabhängige Abfragen verstanden werden. Das Vorgangsbearbeitungsmodul dient insbesondere zur Sammlung aller erforderlichen Daten für einen Vorgang. Das Vorgangsbearbeitungsmodul 30 ist auf einem Computermittel installiert, das in Verbindung mit einer Benutzeroberfläche 10 steht. Die Benutzeroberfläche befindet sich auf einem Anzeigemittel 12. Bei dem Anzeigemittel kann es sich bei-

spielsweise um einen Computermonitor, ein Automatendisplay, das Display eines Mobiltelefons oder eine Website handeln.

Auf dem Computermittel sind vorzugsweise mehrere Vorgangsbearbeitungsmodule installiert. Dies ist schematisch in Fig. 2 dargestellt, in welcher zwei von mehreren Vorgangsbearbeitungsmodulen mit VGA1 und VGA2 bezeichnet sind. Die Vorgangsbearbeitungsmodule können sich dabei im Frontend oder im Backend eines Gesamtsystems aus mehreren Komponenten befinden. Vorzugsweise ist das VGA im Frontend angeordnet. Bei einem durch das Modul abgebildeten Vorgang handelt es sich beispielsweise um eine Transaktion, die von einem Benutzer 20 an einer graphischen Benutzeroberfläche 10 in Verbindung mit dem jeweiligen Vorgangsbearbeitungsmodul 30 abgewickelt wird. Der Benutzer kann beispielsweise ein Mitarbeiter in einer Filiale sein, welcher den Vorgang für einen Kunden durchführt. Der Benutzer kann ferner der Kunde selbst sein, welcher an einem Automaten eine Transaktion tätigt.

Vorzugsweise handelt es sich bei dem Gesamtsystem, in welches die Vorgangsbearbeitungsmodule integriert sind, um ein modulares System, in welchem die Vorgänge bis auf kleinste Funktionseinheiten unterteilt sind. Daher kann ein Vorgangsbearbeitungsmodul auch einen Prozess abbilden, welcher Bestandteil mehrerer anderer Vorgangsbearbeitungsmodule ist. Beispielsweise kann ein VGA das Einscannen eines Barcodes, die Erfassung von Kundendaten oder die Berechnung eines Wertes abbilden. Diese Einzelprozesse können als Module von anderen Vorgangsbearbeitungsmodulen aufgerufen werden, so dass sie nur einmalig implementiert werden müssen. Für jedes Vorgangsbearbeitungsmodul und damit für jeden Vorgang wird eine Benutzeroberfläche bereitgestellt, wobei der modulare

Aufbau den Vorteil hat, dass bei einer Veränderung des jeweiligen Einzelprozesses die Benutzeroberfläche ebenfalls nur einmalig in dem betreffenden Vorgangsbearbeitungsmodul verändert werden muss.

Das Vorgangsbearbeitungsmodul 30 steht über das verwendete Computermittel und ein Anzeigemittel 12 in Verbindung mit der graphischen Benutzeroberfläche 10, die mehrere Steuerelemente 11 aufweist. Bei den Steuerelementen handelt es sich beispielsweise um Schaltflächen oder Eingabefelder, die eigene Funktionalitäten wie beispielsweise Einfeld- oder Mehrfeldplausibilitäten enthalten können. Die Benutzeroberfläche befindet sich in Form eines Anwendungsrahmens vorzugsweise auf einem Client, bei dem es sich um eine Anwendung in einem Netzwerk handelt, welche die Dienste eines oder mehrerer Server in Anspruch nimmt. Der Client initialisiert beim Start den Anwendungsrahmen der graphischen Benutzeroberfläche. Der Client kann dabei eine Plattform mit weiteren Komponenten eines Frameworks wie Datenbanken, Abrechnungsmodule etc. nutzen. In einem besonders bevorzugten Ausführungsbeispiel der Erfindung kann eine als CARBON(**C**omponent **AR**chitecture for **B**usiness **O**ptimizatio**N**)-Plattform bezeichnete Plattform genutzt werden, welche als Software realisiert und auf Datenverarbeitungsanlagen installiert ist. Das Framework beinhaltet dabei die Laufzeitbibliothek der Plattform, welche auch als Common Language Runtime bezeichnet wird (CLR). Nach der Initialisierung des Anwendungsrahmens durch den Client initialisiert und startet dieser zweckmäßigerweise den Kern des Frameworks. Vor der Initialisierung des Kerns können Prüfungen durchgeführt werden, welche beispielsweise die Integrität des Programms betreffen. Ferner kann überprüft werden, ob es sich um einen Start oder einen Recovery-Fall

handelt, welcher durch einen Abbruch eines Vorganges erzeugt wurde.

Das Vorgangsbearbeitungsmodul 30 befindet sich vorzugsweise ebenfalls auf dem Client. Dabei können auf dem Client mehrere verschiedene Versionen eines Vorgangsbearbeitungsmoduls installiert sein. Beim Starten des Vorgangsbearbeitungsmoduls wird automatisch die Version mit der höchsten Versionsnummer herangezogen. Um dies zu ermöglichen, sind die Vorgangsbearbeitungsmodule zweckmäßigerweise in einer bestimmten Weise in einem Vorgangsbearbeitungsmodul-Verzeichnis abgelegt worden.

Die graphische Benutzeroberfläche (GUI) ist vorzugsweise als Assistent-basiertes System konzipiert, bei dem von einem Anwender 20 im Laufe eines Vorganges mehrere Masken in einer vorgegebenen Reihenfolge abgearbeitet werden. Dies hat den Vorteil, dass für einen Vorgang benötigte Daten vollständig erfasst werden und ferner sichergestellt ist, dass der Anwender beim Abschluss des Vorganges alle möglichen Optionen zu Gesicht bekommen hat.

In einem bevorzugten Ausführungsbeispiel der Erfindung ist das Vorgangsbearbeitungsmodul ebenfalls an ein Framework angeschlossen, das vorzugsweise objektorientiert und komponentenorientiert ausgebildet ist. Bei dem Framework handelt es sich um ein API (Application Programming Interface), welches die Funktionalitäten bereitstellt, Clients für verschiedene Assistent-basierte Vorgangsarten zu erstellen. Die von dem Framework zur Verfügung gestellten Funktionalitäten umfassen beispielsweise Kernfunktionalitäten wie das Starten/Beenden des Clients oder der Vorgangsbearbeitungsmodule. Das Framework stellt beispielsweise ein gemein-

sames Startmodul 100 zum Starten der Vorgangsbearbeitungsmodule bereit. Ferner unterstützt das Framework die Vorgangsbearbeitungsmodule bei Datenhaltung und Ablaufsteuerung und stellt Komponenten für den Zugriff auf Daten oder zum Drucken bereit. Darüber hinaus werden Basisfunktionalitäten wie die Protokollierung oder die Fehlerbehandlung bereitgestellt.

Als vollständig objektorientierte Programmiersprache innerhalb des Frameworks kann beispielsweise C# verwendet werden. Die objektorientierte Programmierung sieht Objekt-Klasse-Beziehungen vor, wobei eine Klasse ein abstrakter Oberbegriff für Objekte ist, die eine gemeinsame Struktur und/oder ein gemeinsames Verhalten aufweisen. Ein Objekt ist dabei ein zur Laufzeit eines Programms vorhandenes Exemplar einer Klasse, für das Speicherplatz zur Verfügung gestellt wird. Ein Objekt ist eine Instanz einer Klasse. Eine Klasse und damit ihre Objekte werden durch Attribute, Operationen, Zusicherungen und Beziehungen charakterisiert, wobei die Klassen die Attribute und Methoden zu einer Einheit zusammenfassen. Die Attribute sind standardmäßig nur über Methoden der Klasse zugänglich, so dass eine Kapselung oder Zugriffsbeschränkung vorliegt. Eine Klasse kann ferner keine, eine oder mehrere explizite Schnittstellen besitzen und die Kapselung trennt die Implementierung von der Schnittstelle.

In einem besonders bevorzugten Ausführungsbeispiel der Erfindung ist das Vorgangsbearbeitungsmodul 30 kein eigenständiges Programm, sondern benötigt zu seiner Ausführung die Infrastruktur eines Clients. Es handelt sich jedoch um eine aktive Komponente, welche beispielsweise das eigene Pausieren und/der den Start anderer Vorgangsbearbeitungsmodule bewirken kann.

Das Vorgangsbearbeitungsmodul 30 weist verschiedene Komponenten auf, welche vorzugsweise unabhängig oder abhängig von der Benutzeroberfläche 10 ausgeführt sind. Erfindungsgemäß ist dabei die Steuerung der Benutzeroberfläche unabhängig von Einheiten ausgeführt, welche die fachliche Funktionalität des Vorgangs abbilden und durchführen. Ferner können einzelne Bereiche im Frontend auf dem Client angeordnet sein, während sich andere Bereiche im Backend oder in einem Umsystem befinden können. Die zu einem Vorgangsbearbeitungsmodul gehörende Funktionseinheit 90 ist erfindungsgemäß unabhängig von der graphischen Benutzeroberfläche ausgebildet. In dieser Funktionseinheit befinden sich somit keine Definitionen und Navigationen von Masken, da sonst bei einer Veränderung der Benutzeroberfläche eine Anpassung der Funktionseinheit erfolgen müsste. Ferner hat die Funktionseinheit keinen Zugriff auf GUI-abhängige Instanzen. Werden von dem Vorgangsbearbeitungsmodul Benutzeroberflächen beispielsweise in Form von Web-Frontends unterstützt, kennt der zugehörige Server den Client nur solange, bis eine Anfrage des Clients beantwortet ist. Die Funktionseinheit würde in diesem Fall zweckmäßigerweise im Backend betrieben.

Die Funktionseinheit 90 liegt in einem besonders bevorzugten Ausführungsbeispiel der Erfindung als gekapseltes *BusinessObjekt* vor. Ein Vorgangsbearbeitungsmodul ist dabei genau einem logischen *BusinessObjekt* zugeordnet, das wiederum aus mehreren Klassen mit Verweisen auf Komponenten bestehen kann. Die Aufgabe der Funktionseinheit 90 liegt in der Bereitstellung der fachlichen Funktionalität zur Durchführung eines Vorganges. Durch Aktionen eines Benutzers 20 an der graphischen Benutzeroberfläche 10 wird das *BusinessObjekt* angesprochen. Bei der Aktion an der Benutzeroberfläche kann es sich beispielsweise um das Aktivieren einer Schaltfläche

oder die Änderung eines Wertes in einem Feld handeln. Dabei dienen Operationen als Bindeglied zwischen der Funktionseinheit 90 und den GUI-Controls 11 der Benutzeroberfläche 10. Als Operationen werden dabei Commands wie beispielsweise NavigationCommands, GenericCommands oder VGAStartCommands bezeichnet.

Vorzugsweise werden nicht alle Operationen eines Vorgangsbearbeitungsmoduls in einer Klasse implementiert. Im Sinne einer objektorientierten Kapselung verschiedener Aktionen in Klassen hat es sich als vorteilhaft erwiesen, jede Operation in einer eigenen Klasse zu implementieren. Als Ausnahme können jedoch logisch gleiche Aktionen angesehen werden. Beispielsweise kann der Anwender beim Drücken eines "Weiter"- und des "Cancel"-Buttons gefragt werden, ob diese Aktion wirklich durchgeführt werden soll. In diesem Fall handelt es sich tatsächlich um eine gleiche Aktion, die in einer gemeinsamen Klasse, jedoch mit unterschiedlichen Operationsnamen zum Einsatz kommt.

Das VGA weist ferner einen Datencontainer 40 auf. Der Datencontainer ist ebenfalls unabhängig von der Benutzeroberfläche 10 ausgebildet. Der Datencontainer ist vorzugsweise nach dem Siting-Prinzip einer Container-Site Architektur ausgebildet, wodurch eine einfache Entwicklung ermöglicht wird und Weiterentwicklungen unterstützt werden. Der Datencontainer bildet dabei eine abstrakte generische Schnittstelle von dem Vorgangsbearbeitungsmodul zu Kernkomponenten des Frameworks. Das Framework kommuniziert vorzugsweise über ein Interface *IVGAEntrypoint* mit dem Datencontainer.

Der Datencontainer 40 ist die zentrale Sammelstelle von Daten innerhalb eines Vorgangsbearbeitungsmoduls 30 und ist somit

für die Speicherung des Zustandes des VGA zuständig. Alle relevanten Daten, die in dem VGA anfallen, werden im Datencontainer abgelegt. Dabei kann es sich um von einem Benutzer eingegebene oder vom System ermittelte Daten handeln. Die Daten werden hierbei in Form von Datenobjekten abgelegt, d.h. jedes Datenelement ist einem Datenobjekt zugehörig und kann nur über diese gelesen, geschrieben und geändert werden. Es hat sich als besonders vorteilhaft erwiesen, jedem Vorgangsbearbeitungsmodul 30 einen eigenen Datencontainer 40 zuzuweisen. Dies hat gegenüber der Verwendung einer allgemeinen Datenbank für alle Vorgangsbearbeitungsmodule den Vorteil, dass verschiedene Zusatzfunktionen durchgeführt werden können. Beispielsweise ist beim Abbruch eines Vorgangsbearbeitungsmoduls ein einfacher Neustart in dem Zustand möglich, in welchem der Abbruch stattfand. Da der Zustand des Vorgangsbearbeitungsmoduls vollständig im Datencontainer abgebildet ist, kann der Zustand beim Neustart rekonstruiert und weitergeführt werden, wobei lediglich ein geringer Datenbestandteil gespeichert und rekonstruiert werden muss.

Welche Datenobjekte im Datencontainer 40 existieren und mit welchen Werten diese gefüllt sind, wird erfindungsgemäß in einer Datencontainer-Konfigurationsdatei 61 angegeben. Diese Datei liegt im Verzeichnis des VGA und entspricht einem definierten Schema. Bei der Konfigurationsdatei handelt es sich vorzugsweise um eine Datei im XML-Format. Beim Instanziieren des Datencontainers wird diese Datei geladen, die konfigurierten Datenobjekte werden erstellt und mit den angegebenen Start-Werten initialisiert. Die konfigurierten Datenobjekte stehen dann zur Verfügung und können an ein Steuerelement 11 der graphischen Benutzeroberfläche 10 gebunden und/oder durch die im *BusinessObjekt* definierte Programmlogik gelesen, geschrieben und geändert werden. Ist

der Datencontainer instanziiert, wird diese Instanz dem Konstruktor des *BusinessObjekts* übergeben. Dadurch kann das *BusinessObjekt* auf die Datenobjekte des Datencontainers zugreifen. Der Datencontainer existiert jedoch prinzipiell unabhängig vom *BusinessObjekt*, da dieses erzeugt und wieder vernichtet werden kann.

Ein wesentlicher Bestandteil eines Vorgangsbearbeitungsmoduls 30 ist ferner wenigstens ein Maskenarbeitsbereich 50, der abhängig von der Benutzeroberfläche ausgebildet ist. Beinhaltet ein Vorgangsbearbeitungsmodul einen Maskenfluss aus mehreren Masken, die nacheinander durchlaufen werden, liegen mehrere Maskenarbeitsbereiche vor. Diese Maskenarbeitsbereiche 50 werden im Folgenden auch als VGA-Workspaces bezeichnet, wobei ein Workspace ein technisches Synonym für eine Maske darstellt. Ein VGA-Workspace bildet das Fundament der Steuerelemente 11 (GUI-Controls) der Benutzeroberfläche 10. Bei einem VGA-Workspace handelt es sich um ein spezielles Interface zur Unterstützung der Kommunikation zwischen den Steuerelementen 11 und einer Darstellungskomponente 80, wobei die Darstellungskomponente wiederum in Verbindung mit dem Datencontainer 40 bzw. der Funktionseinheit 90 steht. Wird ein Vorgang eines Vorgangsbearbeitungsmoduls über die Benutzeroberfläche 10 abgewickelt, beinhaltet dies einen Maskenfluss mit verschiedenen Masken, wobei jede Maske durch einen VGA-Workspace beschrieben ist. Jede Maske wird von einer Klasse implementiert, die sich vom VGA-Workspace ableitet. Wird ein Maskenfluss für ein Vorgangsbearbeitungsmodul programmiert, werden die Eigenschaften dieser Klasse entsprechend der Fachlogik eingestellt. In der Fig. 1 sind die verschiedenen VGA-Workspaces 50 für den Maskenfluss des Vorgangsbearbeitungsmoduls 30 beispielhaft mit einem nichtlinearen Maskenfluss dargestellt.

Die Verbindung zwischen den Maskenarbeitsbereichen 50 und der Funktionseinheit 90 bzw. dem Datencontainer 40 wird durch eine Darstellungskomponente 80 hergestellt. Die Darstellungskomponente ist dazu als Zwischenschicht zwischen der Funktionseinheit bzw. dem Datencontainer und den VGA-Workspaces ausgebildet. Die Darstellungskomponente wird im Folgenden auch als Presenter bezeichnet und stellt eine Abstraktionsebene dar. Der Presenter dient der Kommunikation zwischen den Steuerelementen 11 der aktuellen Maske und den zugehörigen Datenobjekten des Datencontainers, wobei der Presenter jedoch nicht auf GUI-Controls zugreift. Der Presenter stellt ferner eine Infrastruktur für die Ausführung von Operationen der Funktionseinheit bereit.

Zu einem Vorgangsbearbeitungsmodul gehören vorzugsweise mindestens drei Dateien. Bei der ersten Datei handelt es sich um ein Assembly *VGAxxxx.dll* des Vorgangsartenmoduls, in welchem die gesamte Vorgangsbearbeitungslogik kodiert sein kann. Das Assembly muss jedoch zumindest die zum Start eines Vorgangsbearbeitungsmoduls notwendigen Klassen implementieren. Die zweite Datei *FormFlow.config* beinhaltet die Konfiguration für die Operationen und den Maskenablaufplan des Vorgangsbearbeitungsmoduls. Die dritte Datei *DataContainer.config* enthält die Konfiguration für die von dem Vorgangsbearbeitungsmodul verwendeten Daten und deren Start-Werte.

Die Kommunikation zwischen den Schichten erfolgt in einem bevorzugten Ausführungsbeispiel der Erfindung über Datenobjekte. Vorzugsweise besitzen alle GUI-Controls 11 die Eigenschaften *NameAspect* und *ValueAspect*. Die *NameAspect*-

Eigenschaft verweist hierbei auf das namensgebende Datenobjekt, während die *ValueAspect*-Eigenschaft auf das wertgebende Datenobjekt verweist. Je nach Control werden ein oder zwei Datenobjekte eingebunden. Wird nur ein Datenobjekt eingebunden, wird automatisch die jeweils andere Eigenschaft auf denselben Eigenschaftswert gesetzt. Ferner können jedoch auch standardmäßige GUI-Controls ohne zugeordnete Datenobjekte verwendet werden.

Das Attributieren der Methoden im *BusinessObjekt* findet mittels Klassen statt. Das Attributieren der Methoden im *BusinessObjekt* ermöglicht die Unabhängigkeit der Namen dieser Methoden von den Namen der Datenobjekte bzw. der Operationen. Eine attributierte Methode kann zu mehreren Datenobjekten oder Operationen aufgerufen werden. Die Konstruktoren der Attribute-Klassen können als Parameter konstante Werte entgegennehmen. Der Presenter ruft bei der Validierung - der Reaktion auf eine Änderung eines Datenobjektes und dem Ausführen einer Operation - jeweils eine zentrale virtuelle Methode am BusinessObjekt auf.

Ein VGA-Workspace verwaltet eine Liste der auf seiner Maske angeordneten GUI-Controls und der jeweils zugeordneten Datenobjekte des Datencontainers 40. Ein Datenobjekt repräsentiert einen Wert, der im Datencontainer gehalten wird, sowie einige ergänzende Parameter. Einem Steuerelement 11, das einen bestimmten Wert darstellen soll, wird zum Designzeitpunkt daher nur das entsprechende Datenobjekt des Datencontainers zugewiesen. Zur Laufzeit des Systems wird aufgrund dieses Datenobjektes eine Beziehung des GUI-Controls zu dem darzustellenden Datenobjekt hergestellt, der im Datencontainer gehalten wird. Ändert sich innerhalb des *BusinessObjekts* ein Wert, der auf der Benutzeroberfläche

dargestellt werden soll, dann wird das betreffende Datenobjekt, das den Wert enthält, über den Presenter an die VGA-Maske weitergeleitet. Dies gilt ebenso für alle GUI-Controls, welche dieses Datenobjekt darstellen.

Eine andere Form der oben genannten Kommunikation findet statt, wenn durch eine Benutzereingabe eine Wertänderung auf der Benutzeroberfläche stattgefunden hat. In diesem Fall muss das durch die Eingabe aktualisierte GUI-Control den neuen Wert vorzugsweise nach einer internen Plausibilitätsprüfung an das *BusinessObjekt* propagieren. Nach erfolgreicher Validierung auf der Operationenebene und dann auf der Ebene der Funktionseinheit wird der neue Wert im betreffenden Datenobjekt des Datencontainers gespeichert.

Die Prüfung eingegebener Werte erfolgt somit nicht auf der GUI-Ebene, sondern im Bereich des Datencontainers. Beispielsweise ist es für die GUI-Controls und die VGA-Workspaces nicht erforderlich, dass diese eine Überprüfung von Eingabewerten beispielsweise auf die Zeichenlänge oder einen Datumsbereich beinhalten. Darf ein Eingabewert beispielsweise nur eine Länge von 15 Zeichen haben, wird dies nicht durch die GUI-Control überprüft, sondern der Wert wird dem entsprechenden Datenobjekt im Datencontainer zugeordnet. Anhand der Eigenschaften des Datenobjektes wird überprüft, ob der Wert die durch den Datencontainer gesetzte Voraussetzung von 15 Zeichen erfüllt oder nicht. Ist dies der Fall, wird der Wert angenommen und im Datencontainer gespeichert. Wird die Voraussetzung nicht erfüllt, erfolgt eine Meldung an den Benutzer 20. Die Trennung der GUI-Controls von der Funktionseinheit und dem Datencontainer hat hierbei den Vorteil, dass die GUI-Controls und damit die Benutzeroberfläche nicht umprogrammiert werden müssen, falls sich beispielsweise die

Voraussetzung für einen Eingabewert ändern sollen. In diesem Fall wird lediglich die Datencontainer-Konfigurationsdatei 61 *DataContainer.config* geändert, mit welcher der Datencontainer initiiert und instanziiert wird.

Das Vorgangsbearbeitungsmodul 30 kann vorzugsweise auf mehrere Arten gestartet werden. Beispielsweise kann der Start über einen Menü-Button auf der Benutzeroberfläche oder aus einem anderen Vorgangsbearbeitungsmodul heraus gestartet werden. Für den Start eines Vorgangsbearbeitungsmoduls ist ein Startmodul 100 zuständig. Das Startmodul kann auch als *ContextManager* bezeichnet werden, der Teil eines *ApplicationManagers* des Frameworks ist. Der Start wird vorzugsweise durch eine Klasse *VGABase* unterstützt, welche alle Funktionalitäten zum Starten eines Vorgangsbearbeitungsmoduls beinhaltet. Die Klasse *VGABase* stellt eine abstrakte Basisklasse für alle Datencontainer-Klassen dar. Sie implementiert insbesondere die *IVGAEntryPoint*-Schnittstelle.

Beim Start eines VGA instanziiert und initialisiert das Startmodul 100 den Datencontainer 40 und die Funktionseinheit 90 des betreffenden Vorgangsbearbeitungsmoduls. Ferner wird ein Controller 70 instanziiert, initialisiert und gestartet. Der Controller wird dabei von der Klasse VGABase erzeugt und initialisiert. Der Controller 70 hat die Aufgabe, die Abfolge der verschiedenen VGA-Workspaces 50 zu steuern. Der Controller sorgt ferner dafür, dass ein Vorgangsbearbeitungsmodul korrekt gestartet und beendet wird und die jeweils richtigen VGA-Workspaces angezeigt werden.

Beim Initialisieren des Datencontainers 40 werden die Datenobjekte angelegt und mit den Start-Werten initialisiert,

welche aus einer Datencontainer-Konfigurationsdatei 61 *DataContainer.config* ausgelesen werden. Die Datencontainer-Konfigurationsdatei 61 enthält die Konfiguration der von dem Vorgangsbearbeitungsmodul verwendeten Daten und deren Startwerte. Beim Initialisieren des Controllers 70 wird ein Maskenablaufplan des Vorgangsbearbeitungsmoduls 30 erstellt. Dazu liest der Controller die Konfiguration des Maskenflusses aus einer Controller-Konfigurationsdatei 60 *FormFlow.config* aus. Dazu kann eine Klasse *FormFlowReader* verwendet werden, welche den Maskenablaufplan des Vorgangsbearbeitungsmoduls aus der *FormFlow.config* ausliest. Die Klasse *FormFlowReader* wird beim Start eines VGA vom Controller aufgerufen. Die Controller-Konfigurationsdatei 60 beinhaltet neben der Konfiguration des Maskenablaufplanes auch die Operationen des Vorgangsbearbeitungsmoduls.

Die Controller-Konfigurationsdatei 60 liegt vorzugsweise im XML-Format vor und enthält Informationen über den Maskenablaufplan eines Vorgangsbearbeitungsmoduls, sowie über die zu dem jeweiligen Vorgangsbearbeitungsmodul gehörenden Operationen. Die Konfiguration wird gegen ein Schema validiert und bei erfolgreicher Validierung erstellt der Controller den Maskenablaufplan. Daraufhin erzeugt der Controller eine Darstellungskomponente 80 und initialisiert diese. Bei jedem Maskenwechsel wird die Darstellungskomponente vom Controller neu initialisiert. Ferner ermittelt der Controller den ersten VGA-Workspace in der Maskenreihenfolge, initialisiert diesen und zeigt ihn an. Der Controller verwendet für die Maskennavigation einen Stapel, der beim Start eines Vorgangsbearbeitungsmoduls erzeugt wird. Die Maske des ersten VGA-Workspaces wird als erste auf den Stapel gelegt. Falls ein Vorgangsbearbeitungsmodul auf der letzten Maske gestartet werden soll, so werden alle Masken auf den Stapel

gelegt, so dass die letzte Maske zur obersten Maske auf dem Stapel wird. Der VGA-Workspace 50 initialisiert daraufhin die Steuerelemente 11 auf seiner Oberfläche, wobei er über die Darstellungskomponente 80 auf die Werte in den zugeordneten Datenobjekten des Datencontainers 40 zugreift.

Bei der Initialisierung geht der VGA-Workspace alle auf seiner Oberfläche angeordneten GUI-Controls durch und ermittelt für jedes Control mit Hilfe der Eigenschaft *AspectName*, welchem Datenobjekt es zugeordnet ist. Anschließend ruft der VGA-Workspace den aktuellen Wert jedes einzelnen Datenobjektes über den Presenter ab. Dies geschieht beispielsweise über eine Methode *GetAspect()* am Presenter, der der Datenobjektname als Parameter übergeben wird. Daraufhin delegiert der Presenter den Methodenaufruf an das *BusinessObjekt*, welches das entsprechende Datenobjekt aus dem Datencontainer abruft.

Ist für ein Vorgangsbearbeitungsmodul 30 ein linearer Maskenfluss vorgesehen, kann das VGA auf der ersten oder der letzten Maske gestartet werden. Ist ein nicht-linearer Maskenfluss vorgesehen, wird das VGA standardmäßig auf der ersten Maske gestartet, da im nicht-linearen Maskenfluss der Rückweg nicht eindeutig definiert ist. Auf welche Maske innerhalb des Maskenflusses navigiert wird, wird durch Navigationsbefehle in den Operationen festgelegt. Die Navigationsbefehle sind in der Controller-Konfigurationsdatei 60 enthalten und sind somit veränderbar. Dabei wird jedoch nicht direkt angegeben, auf welche Maske navigiert werden soll, sondern nur welcher Weg von der aktuellen Maske zu wählen ist, wodurch sich die Folgemaske implizit ergibt. Alle möglichen Wege müssen in der Controller-Konfigurationsdatei

60 *FormFlow.config* festgelegt sein. Durch diese Vorgehensweise ist es im Nachhinein möglich, Masken aus dem Maskenfluss zu entfernen (Ändern der Wege in der Konfiguration), ohne unbedingt die Operationen der Funktionseinheit anpassen zu müssen.

Die einzelnen Wege zu der nächsten Maske werden beispielsweise mit IDs (Zeichenketten) versehen. Die ID „next" wird dabei beispielsweise durch das Framework definiert und gibt den Standardweg beim Verlassen einer Maske an. Soll dieser Weg beschritten werden, so geschieht dies durch diesen Standard-Navigationsbefehl automatisch, ohne dass ein Befehl implementiert werden muss. Eine Implementierung ist daher nur erforderlich, wenn einer von mehreren Wegen beschritten werden soll.

Mittels Navigationsbefehle navigiert der Controller 70 zum nächsten oder vorherigen VGA--Workspace. Wenn der Controller mit Hilfe des Ablaufplans die nächste Maske ermittelt, legt er sie auf den Stapel. Bei einem umgekehrten Maskenwechsel in die Rückwärts-Richtung nimmt der Controller die oberste Maske vom Stapel. Ist also die erste Maske die aktuelle, liegen keine Masken auf dem Stapel.

Liegt ein nicht-linearer Maskenablauf vor, kann es Masken geben, die über mehrere alternative Nachfolgemasken verfügen. Die Klasse *VGAController* ist beispielsweise dafür zuständig, ausgehend von dem aktuellen VGA-Workspace den folgenden VGA-Workspace zu identifizieren, ihn zu erzeugen, zu initialisieren und anzuzeigen. Dabei wird bei jedem Maskenwechsel der Presenter vom Controller erzeugt und diesem wird der aktuelle VGA-Workspace mit den zugehörigen Operationen übergeben.

Damit der Presenter Methoden des *BusinessObjekts* aufrufen kann, gibt der Controller die Referenz auf das *BusinessObjekt* an den Presenter weiter. Der aktuelle Status des Clients kann in Form eines technischen oder fachlichen Checkpoints weggeschrieben werden. Damit ein Vorgangsbearbeitungsmodul gesichert und wiederhergestellt werden kann, muss sichergestellt sein, dass alle Daten, die innerhalb des VGA von Belang sind, in einem Datenobjekt abgelegt sind. Dadurch ist gewährleistet, dass bei dem Wiederaufsatz auf einem bestimmten Checkpoint die Werte innerhalb der Datenobjekte wiederhergestellt werden. Dies wird durch die Unterstützung von Serialisierung erreicht. Unter einer Serialisierung versteht man eine sequentielle Abbildung von Objekten auf eine externe Darstellungsform. Ziel ist das Erreichen von Persistenz für das Objekt. Ein Objekt kann zum Beispiel serialisiert werden, indem der komplette Zustand des Objektes, inklusive aller Objekte die es referenziert, in einen Datenstrom umgewandelt und auf ein Speichermedium geschrieben wird. Der Zustand und Inhalt des Datencontainers kann so jederzeit in einen binären Datenstrom umgewandelt und persistiert werden. Somit können alle Daten eines VGA, die in einem Datenobjekt und damit im Datencontainer vorhanden sind, im Falle eines Systemabsturzes wiederhergestellt werden.

Ein Vorgangsbearbeitungsmodul kann vorzugsweise zu jedem Zeitpunkt vom Benutzer 20 abgebrochen werden. Dies erfolgt beispielsweise über das Betätigen eines Abbrechen-Buttons auf der aktuellen Maske. Im Gegensatz dazu erfolgt das ordnungsgemäße Beenden eines Vorgangsbearbeitungsmoduls nur über den Weiter-Button auf der letzten Maske.

Checkpoints können beispielsweise standardmäßig bei jedem Maskenwechsel in die Vorwärts-Richtung vom Controller

weggeschrieben werden. Zusätzlich können an bestimmten Stellen im Maskenarbeitsbereich weitere Checkpoints vorgesehen sein. In einem besonders bevorzugten Ausführungsbeispiel der Erfindung wird ferner bei jedem Maskenwechsel überprüft, ob alle Pflichtfelder der Maske bearbeitet wurden. Die Aktion, mit welcher der Pflichtfeld-Eigenschaft eines Steuerelementes 11 Genüge getan ist, hängt dabei von dem jeweiligen Steuerelement und seinem zugeordneten Datenobjekt ab. Bei erfolgloser Pflichtfeld-Validierung werden die Pflichtfelder vorzugsweise so visualisiert, dass der Benutzer 20 sofort erkennen kann, dass in den betreffenden Feldern noch Eingaben zu tätigen sind.

In einem weiteren besonders bevorzugten Ausführungsbeispiel der Erfindung ist es ferner möglich, eine Befüllung eines Datencontainers eines Vorgangsbearbeitungsmoduls durchzuführen, ohne dass GUI-Elemente einer Benutzeroberfläche real befüllt und abgearbeitet werden müssen. Dies ist beispielsweise vorteilhaft, wenn Vorrichtungen in das Gesamtsystem eingebunden werden, auf denen ein Vorgangsbearbeitungsmodul oder ein Maskenarbeitsbereich nicht lauffähig sind. So ist es möglich, beliebige Vorrichtungen in das Gesamtsystem zu integrieren und von den Vorrichtungen erzeugte Daten zu verarbeiten. Sollen beispielsweise Daten von Automaten verarbeitet werden, auf denen ein Vorgangsbearbeitungsmodul für den durch den Automaten durchgeführten Vorgang nicht installiert ist, erzeugt der Automat eine Datei, welche dem System übermittelt wird. Die Datei kann über eine entsprechende Schnittstelle eingelesen und damit ein Vorgangsbearbeitungsmodul für den betreffenden Vorgang des Automaten gestartet werden. Das Vorgangsbearbeitungsmodul wird durchlaufen und die Datenobjekte des Datencontainers werden automatisch mit Werten der eingespeisten Datei befüllt.

Bezugszeichenliste:

10	Graphische Benutzeroberfläche, GUI
11	Steuerelement, GUI-Control
12	Anzeigemittel, Monitor
20	Benutzer, Anwender
30	Vorgangsbearbeitungsmodul, VGA1, VGA2
40	Datencontainer, DC
41	Datenobjekt
50	Maskenarbeitsbereich, MA, VGA-Workspace
60	Controller-Konfigurationsdatei
61	Datencontainer-Konfigurationsdatei
70	Controller, C
80	Darstellungskomponente, DM, Presenter
90	Funktionseinheit, FE
100	Startmodul, SM

Patentansprüche:

1. Verfahren zum rechnergestützten Erzeugen einer graphischen Benutzeroberfläche (10) für ein Vorgangsbearbeitungsmodul (30), wobei die Benutzeroberfläche wenigstens ein Steuerelement (11) aufweist und das Vorgangsbearbeitungsmodul (30) auf einem Computermittel installiert ist,
gekennzeichnet
durch folgende Schritte:
- Zuordnen von wenigstens einem Maskenarbeitsbereich (50) mit wenigstens einem Steuerelement (11) zu einem Vorgangsbearbeitungsmodul (30), wobei jedem Steuerelement (11) ein Datenobjekt (41) zugeordnet wird;
- Starten des Vorgangsbearbeitungsmoduls (30) durch ein Startmodul (100), wobei beim Starten wenigstens ein Controller (70), eine Funktionseinheit (90) und ein Datencontainer (40) des Vorgangsbearbeitungsmoduls (30) initialisiert werden;
- Einlesen einer Controller-Konfigurationsdatei (60) zur Initialisierung des Controllers (70), wobei die Controller-Konfigurationsdatei (60) die Konfiguration der Darstellung des Maskenarbeitsbereiches (50) des betreffenden Vorgangsbearbeitungsmoduls (30) beinhaltet;
- Einlesen einer Datencontainer-Konfigurationsdatei (61) zur Initialisierung des Datencontainers (40), wobei die Datencontainer-Konfigurationsdatei (61) die Konfiguration der Datenobjekte (41) des Datencontainers (40) des betreffenden Vorgangsbearbeitungsmoduls (30) und die Konfiguration der Operationen für die Kommunikation zwischen dem

Maskenarbeitsbereich (50), der Funktionseinheit (90) und dem Datencontainer (40) beinhaltet;

- Initialisieren einer Darstellungskomponente (80) durch den Controller (70);
- Ermitteln und Initialisieren des Maskenarbeitsbereiches (50) des Vorgangsbearbeitungsmoduls (30) durch den Controller (70); und
- Initialisieren der Steuerelemente (11) des Maskenarbeitsbereiches (50).

2. Verfahren nach Anspruch 1, **dadurch gekennzeichnet,** dass die Steuerelemente (11) als graphische Benutzeroberfläche auf einem Anzeigemittel (12) dargestellt werden und zur Interaktion eines Benutzers (20) mit dem Vorgangsbearbeitungsmodul (30) dienen und dass die Steuerelemente (11) des Maskenarbeitsbereiches (50) gefüllt und/oder abgearbeitet werden, wobei der Maskenarbeitsbereich (50) über die Darstellungskomponente (80) auf die Datenobjekte (41) des Datencontainers (40) zugreift.

3. Verfahren nach Anspruch 1, **dadurch gekennzeichnet**, dass über eine Schnittstelle eine Datenquelle eingelesen wird, deren Inhalt den Datenobjekten (41) des Datencontainers (40) zugeordnet werden kann und dass die Steuerelemente (11) des Maskenarbeitsbereiches (50) gefüllt und/oder abgearbeitet werden, wobei der Maskenarbeitsbereich (50) über die Darstellungskomponente (80) auf die Datenobjekte (41) des Datencontainers (40) zugreift und die Datenobjekte (41) automatisch mit Werten der Datenquelle gefüllt werden.

4. Verfahren nach Anspruch 3,
dadurch gekennzeichnet,
dass es sich bei der Datenquelle um eine Datei handelt.

5. Verfahren nach einem oder mehreren der Ansprüche 1 bis 4,
dadurch gekennzeichnet,
dass bei einer Änderung eines Wertes innerhalb der Funktionseinheit (90) eine Weiterleitung des dem Wert zugeordneten Datenobjektes (41) über die Darstellungskomponente (80) an den Maskenarbeitsbereich (50) und damit alle Steuerelemente (11) erfolgt, welche den Wert darstellen.

6. Verfahren nach einem oder mehreren der Ansprüche 1 bis 5,
dadurch gekennzeichnet,
dass bei einer Änderung eines Wertes eines Steuerelementes (11) eine Weiterleitung des Wertes an die Funktionseinheit (90) und den Datencontainer (40) erfolgt, wobei der Wert zu dem Datenobjekt (41) gespeichert wird, welches dem Steuerelement (11) zugeordnet ist.

7. Verfahren nach einem oder mehreren der Ansprüche 1 bis 6,
dadurch gekennzeichnet,
dass auf dem Computermittel mehrere Vorgangsbearbeitungsmodule (30) installiert werden und dass ein erstes Vorgangsbearbeitungsmodul ein zweites Vorgangsbearbeitungsmodul aufruft, indem das gemeinsame Startmodul (100) das zweite Vorgangsbearbeitungsmodul startet und der Maskenarbeitsbereich des zweiten Vorgangsbearbeitungsmoduls initialisiert wird, wobei der zweite Maskenarbeitsbereich bei der Initialisierung und Abarbeitung der

Steuerelemente über seine Darstellungskomponente auf Inhalte seines Datencontainers zugreift.

8. Verfahren nach einem oder mehreren der Ansprüche 1 bis 7, **dadurch gekennzeichnet**, dass die Controller-Konfigurationsdatei (60) und/oder die Datencontainer-Konfigurationsdatei (61) im XML-Format bereitgestellt werden.

9. Vorrichtung zum rechnergestützten Erzeugen einer graphischen Benutzeroberfläche (10) für ein Vorgangsbearbeitungsmodul (30), wobei die Benutzeroberfläche wenigstens ein Steuerelement (11) aufweist und das Vorgangsbearbeitungsmodul (30) auf einem Computermittel installiert ist,
gekennzeichnet
durch folgende Merkmale:

- Vorgangsbearbeitungsmodul (30) umfassend wenigstens
  - einen Maskenarbeitsbereich (50)
  - einen Controller (70)
  - einen Datencontainer (40) und
  - eine Funktionseinheit (90);
- Startmodul (100) zum Starten des Vorgangsbearbeitungsmoduls (30), wobei das Startmodul Mittel zum Initialisieren wenigstens des Controllers (70), der Funktionseinheit (90) und des Datencontainers (40) aufweist;
- Controller (70) umfassend wenigstens
  - Mittel zum Einlesen einer Controller-Konfigurationsdatei (60),
  - Mittel zum Initialisieren einer Darstellungskomponente (80),

- Mittel zum Ermitteln und Initialisieren des Maskenarbeitsbereiches (50) des Vorgangsbearbeitungsmoduls (30);

- Maskenarbeitsbereich (50) umfassend wenigstens
  - Mittel zum Initialisieren der Steuerelemente (11) des Maskenarbeitsbereiches (50),
  - Mittel zum Zugreifen auf Datenobjekte (41) des Datencontainers (40) über die Darstellungskomponente (80).

10. Vorrichtung nach Anspruch 9,
dadurch gekennzeichnet,
dass das Computermittel in Verbindung mit einem Anzeigemittel (12) zum Darstellen der graphischen Benutzeroberfläche steht und der Maskenarbeitsbereich (50) Mittel zum Darstellen der Steuerelemente (11) auf dem Anzeigemittel (12) aufweist.

11. Vorrichtung nach Anspruch 9,
dadurch gekennzeichnet,
dass die Vorrichtung eine Schnittstelle zum Einlesen einer Datenquelle und zum Zuordnen des Inhaltes der Datenquelle zu Datenobjekten (41) des Datencontainers (40) des Vorgangsbearbeitungsmoduls (30) umfasst und dass die Vorrichtung Mittel zur Befüllung der Datenobjekte (41) mit Werten der Datenquelle aufweist.

12. Vorrichtung nach einem oder mehreren der Ansprüche 9 bis 11,
dadurch gekennzeichnet,
dass auf dem Computermittel mehrere Vorgangsbearbeitungsmodule (30) installiert sind, welche jeweils wenigstens eine Controller-Konfigurationsdatei (60), eine

Datencontainer-Konfigurationsdatei (61), einen Maskenarbeitsbereich (50), eine Funktionseinheit (90) und einen Datencontainer (40) aufweisen, wobei ein Vorgangsbearbeitungsmodul (30) jeweils einen fachlichen Vorgang eines Retailsystems abbildet.

13. Computerlesbares Medium, das Befehle zum Ausführen der Schritte des Verfahrens 1 auf einem Computermittel aufweist.

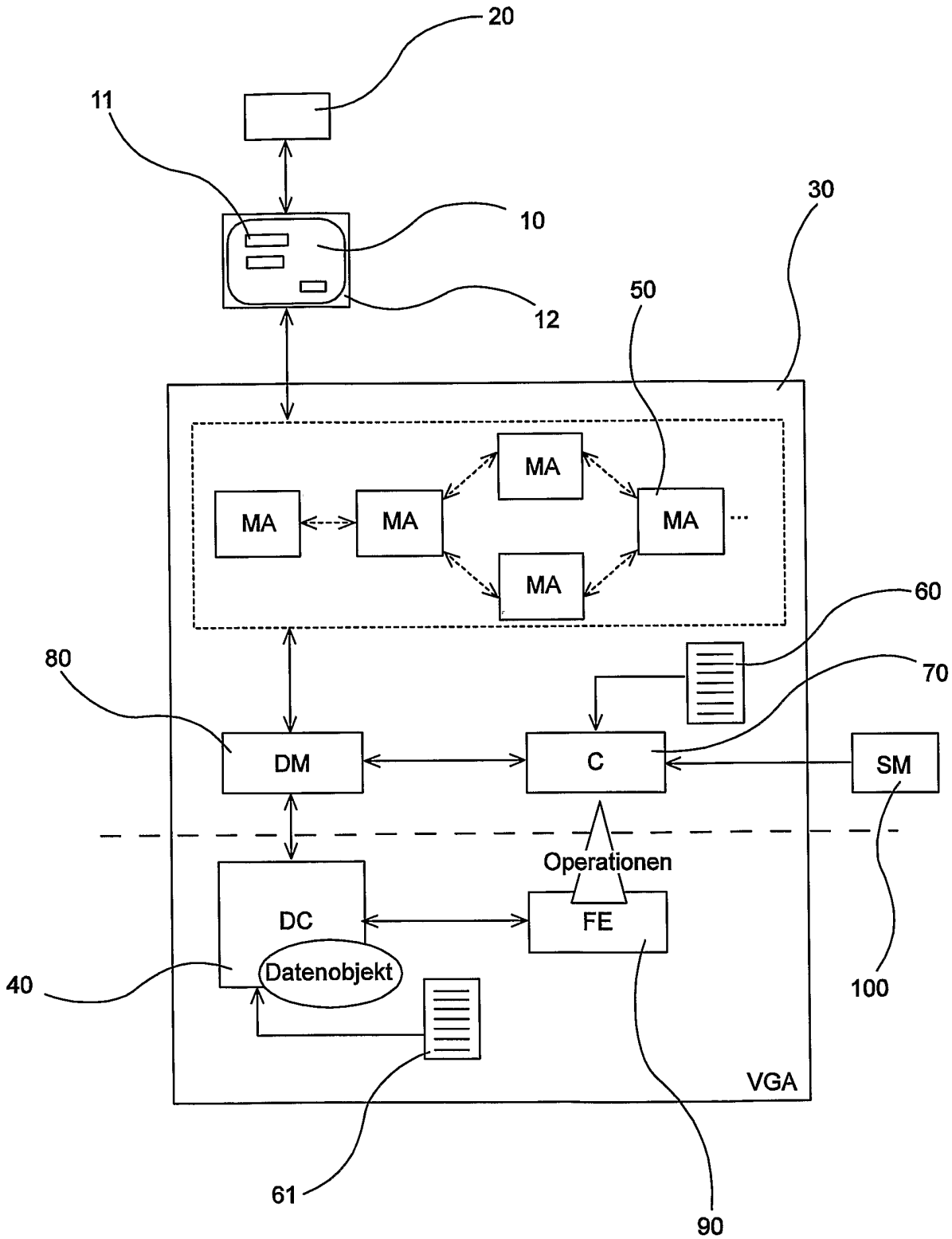

20
11
10
12
30
50
MA
MA
MA
MA
MA
60
80
70
DM
C
SM
Operationen
DC
FE
Datenobjekt
40
100
VGA
61
90

Fig. 1

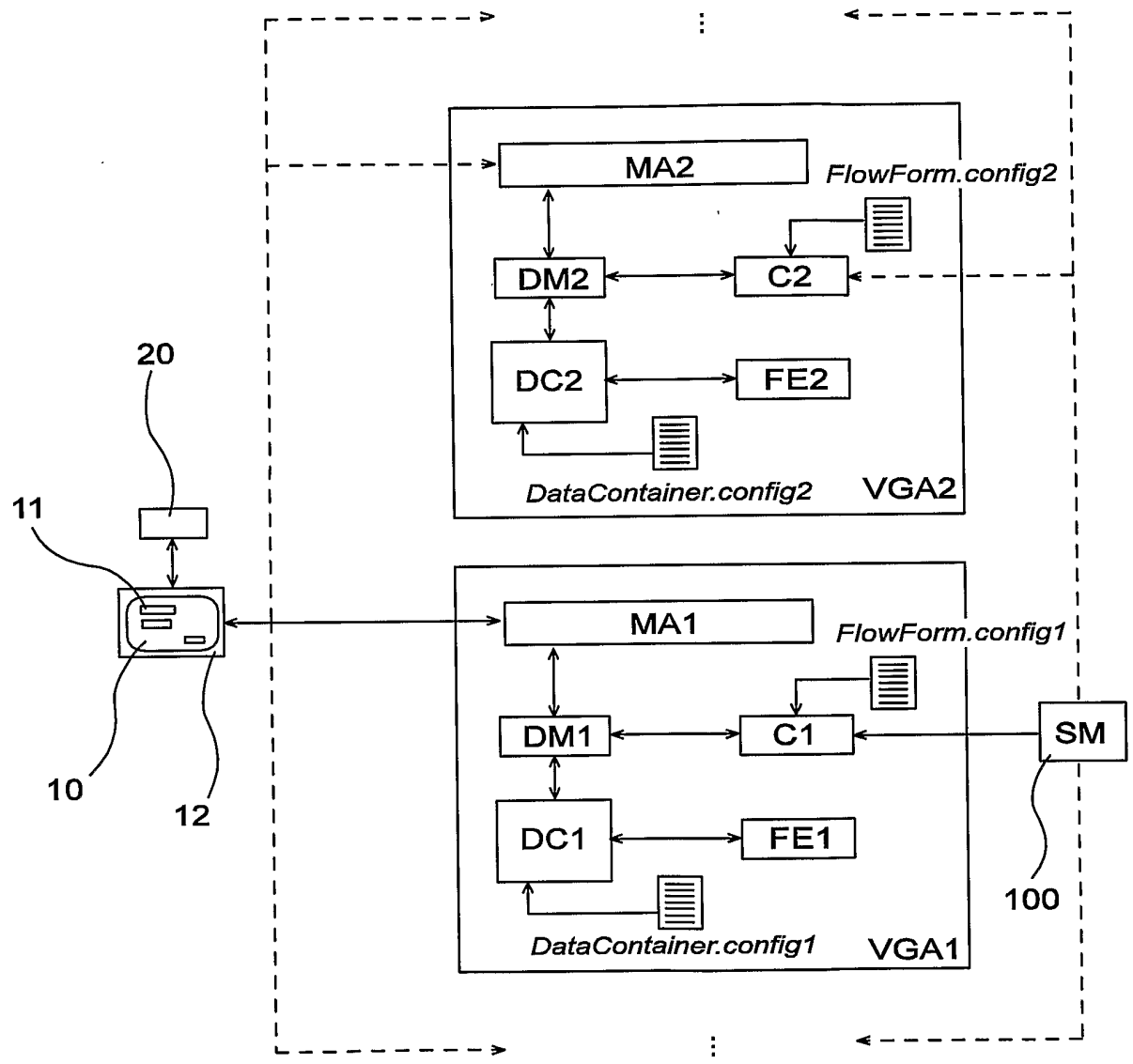

MA2
FlowForm.config2
DM2
C2
DC2
FE2
DataContainer.config2
VGA2
20
11
10
12
MA1
FlowForm.config1
DM1
C1
SM
DC1
FE1
100
DataContainer.config1
VGA1

Fig. 2

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/000841

A. CLASSIFICATION OF SUBJECT MATTER
G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, IBM-TDB, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	-/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
3 April 2006	13/04/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Uhlmann, N

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/000841

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	PFISTERER CHRISTOPH: "A Semantic Description Language for Platform-Independent Graphical User Interfaces" [Online] 2003, , STUTTGART , XP002348223 Retrieved from the Internet: URL:http://elib.uni-stuttgart.de/opus/volltexte/2003/1314/pdf/DIP-2017.pdf> [retrieved on 2005-09-30] page 26, lines 10-22 page 71, line 29 - page 72, line 9 page 78, lines 16-31 page 84, lines 24-30 page 86, line 32 - page 87, line 6 page 88, lines 14-24 page 99, lines 2-5 paragraph [2.2.6] paragraph [03.2] paragraph [5.5.2] paragraphs [06.2] - [6.3.2] paragraph [9.2.8]	1-13
A	WO 2004/107200 A (MICROSOFT CORPORATION) 9 December 2004 (2004-12-09) cited in the application the whole document	1,13
A	HARMONIA INC: "Java Renderer User Manual"[Online] 29 June 2001 (2001-06-29), pages 1-117, XP002348239 Retrieved from the Internet: URL:defense.harmonia.com/products/java/manual/v1.0b/uiml2jr-UserMan.pdf> [retrieved on 2005-09-30] page 69 - page 71 page 88 - page 96	1
A	DE 199 46 660 A1 (SIEMENS AG) 12 April 2001 (2001-04-12) the whole document	1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2006/000841

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2004107200 A	09-12-2004	AU 2003239503 A1	21-01-2005
		BR 0307080 A	08-03-2005
		CA 2470177 A1	17-11-2004
		MX PA04007409 A	17-02-2005
DE 19946660 A1	12-04-2001	NONE	

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP2006/000841

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
G06F9/44

Nach der Internationalen Patentklassifikation (IPC) oder nach der nationalen Klassifikation und der IPC

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
G06F

Recherchierte, aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)
EPO-Internal, WPI Data, IBM-TDB, INSPEC

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
	-/--	

[X] Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen [X] Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :
"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist
"E" älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist
"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)
"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht
"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist
"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist
"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden
"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist
"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
3. April 2006	13/04/2006

Name und Postanschrift der Internationalen Recherchenbehörde	Bevollmächtigter Bediensteter
Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Uhlmann, N

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP2006/000841

C. (Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	PFISTERER CHRISTOPH: "A Semantic Description Language for Platform-Independent Graphical User Interfaces" [Online] 2003, , STUTTGART , XP002348223 Gefunden im Internet: URL:http://elib.uni-stuttgart.de/opus/volltexte/2003/1314/pdf/DIP-2017.pdf> [gefunden am 2005-09-30] Seite 26, Zeilen 10-22 Seite 71, Zeile 29 - Seite 72, Zeile 9 Seite 78, Zeilen 16-31 Seite 84, Zeilen 24-30 Seite 86, Zeile 32 - Seite 87, Zeile 6 Seite 88, Zeilen 14-24 Seite 99, Zeilen 2-5 Absatz [2.2.6] Absatz [03.2] Absatz [5.5.2] Absätze [06.2] - [6.3.2] Absatz [9.2.8]	1-13
A	WO 2004/107200 A (MICROSOFT CORPORATION) 9. Dezember 2004 (2004-12-09) in der Anmeldung erwähnt das ganze Dokument	1,13
A	HARMONIA INC: "Java Renderer User Manual"[Online] 29. Juni 2001 (2001-06-29), Seiten 1-117, XP002348239 Gefunden im Internet: URL:defense.harmonia.com/products/java/manual/v1.0b/uiml2jr-UserMan.pdf> [gefunden am 2005-09-30] Seite 69 - Seite 71 Seite 88 - Seite 96	1
A	DE 199 46 660 A1 (SIEMENS AG) 12. April 2001 (2001-04-12) das ganze Dokument	1

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen
PCT/EP2006/000841

Im Recherchenbericht angeführtes Patentdokument	Datum der Veröffentlichung	Mitglied(er) der Patentfamilie	Datum der Veröffentlichung
WO 2004107200 A	09-12-2004	AU 2003239503 A1	21-01-2005
		BR 0307080 A	08-03-2005
		CA 2470177 A1	17-11-2004
		MX PA04007409 A	17-02-2005
DE 19946660 A1	12-04-2001	KEINE	