

申請日期	90.6.15
案 號	90114791
類 別	G067 9/06

公告本

521210

(以上各欄由本局填註)

發明專利說明書		
一、發明 名稱	中 文	資訊系統、模組及其方法
	英 文	Systeme informatique modulaire et procede associe
二、發明 創作人	姓 名	艾馬迪 Marty Emmanuel
	國 籍	法國
	住、居所	法國蒙特皮爾市瑪瑞可路 13 號 13 rue du Mareschal, 34000 MONTPELLIER, France
三、申請人	姓 名 (名稱)	立可威方案公司 Nexwave Solutions
	國 籍	法國
	住、居所 (事務所)	法國蒙特皮爾市楚思頓碼頭 130-140 號 Immeuble le Seranne, BAT A, 130-140 quai Flora Tristan, 34070 MONTPELLIER, France
	代 表 人 姓 名	安德瑞 (ANDR'E)

裝

訂

線

(由本局填寫)

承辦人代碼：
大類：
I P C 分類：

A6

B6

本案已向：

法國(地區) 申請專利，申請日期： 案號： ，☒有 ☐無主張優先權
西元 2000 年 6 月 16 日 0007754

有關微生物已寄存於： ，寄存日期： ，寄存號碼：

(請先閱讀背面之注意事項再填寫本頁各欄)

裝

訂

線

- 2-2 -

經濟部智慧財產局員工消費合作社印製

五、發明說明(1)

發明背景

一般言之，本發明與允許使用者與電腦硬體互動的軟體系統有關。

更明確地說，本發明與軟體系統有關，其所包括的作業系統能提供彈性以及即使現今最強大的作業系統都無法想像的執行速度等特性。

本發明也與這類系統相關的方法有關。

此外，如以下所見，本發明有助於(非只限於)用來發展與內建式電腦系統相關的應用軟體。

10 吾人回想，作業系統是任何軟體系統中最必需的單元，它構成與電腦系統之硬體組成間直接互動的部分。

現為吾人所知的作業系統極多，這些作業系統允許發展團隊用它來設計應用軟體，並允許終端使用者使用這些應用軟體。

15 現今的作業系統可概略分成兩大系列。

➤ 單體結構(Monolithic architecture)

第一系列是由單體結構的作業系統構成。此類系統構成單一的軟體組，它包括3個主要的層：

- 20 所謂的底層，它對應於系統的核心。可將核心定義成一被動且擁有特權的碼組，它永遠被載入系統的記憶體，且能被所有的套裝應用軟體使用。它的基本任務是調節系統之共享資源(如記憶體、處理器、周邊)的使用。為完成此任務，核心提供某些重要的機能，包括：

五、發明說明 (2)

- ✓ 記憶體的管理(在應用軟體間分配記憶體資源，係依應用軟體之請求等的功能)，
- ✓ 程序的管理(排程、存取處理器的管理、多工作業的管理等)，
- 5 ✓ 周邊之驅動器的管理，
- ✓ 檔案管理(名稱、位址等)，
- ✓ 使用者管理(存取、側寫等)。
- 所謂的頂層，為應用軟體專用，使用者可與其互動，以便將指令輸入系統，以及輸入前往核心的請求，
- 10 • 所謂"程式庫"的中間層，有時將其視為一層，週期性發生的功能集中在其中，以便能快速實施這些功能。

本文的附圖1概略地表示此種單體結構，圖中顯示
15 K(核心)、L(程式庫)及A(應用軟體)的中央集權與階層的配置。

"層"的定義使得作業系統之各單元間的通信能層級化，最底層被定義成"特權"層，可被較高階的層呼叫。

因此，層是軟體單元的分類，在同一層中的軟體單元
20 享有同等的特權。此外，特權可被定義成使用電腦系統某資源(硬體資源或軟體資源)的能力。

如前所述，按照此單體結構所建構的系統組成一單一的軟體組，它是"區塊式"(blockwise)相容。

一般言之，此結構的優點是系統很穩定且快速。

五、發明說明(3)

不過，此類作業系統在系統的彈性與升級性方面呈現重大的限制。

事實上，有系統地修改核心涉及完全重新編譯核心。因此，每次的修改，都需要再次從事系統的總體接受性。

- 5 因此，缺乏彈性對系統而言是不利的，此致使系統的升級既麻煩又複雜，且維持費用昂貴。此外，由於連續的"累加"，可能會使系統變得不穩定。

- 此外，至於內建的電腦系統(on-board system)，意指至少有部分會與機動式的硬體整合，例如車輛、行動電話等，
10 缺乏彈性更是極度地不利，因為這使得電腦系統的某些特性需要事先固定(例如執行的速度、記憶體的大小等)。

事實上，程式設計師經常面對內建應用軟體對相關電腦系統之各種要求間的差異，因此，能自由地定義與升級系統的主要特性乃極有助益。

- 15 因此，由於前文暗示的缺點主要是源自於單體結構的作業系統缺乏彈性，此單體結構基本上意欲用於不需電腦軟體或硬體做重大升級的應用軟體，因此，它非常不適合內建的系統。

20 ➤ 微核心結構

為鬆綁此限制，為作業系統設計出第二種結構。這些系統是微核心式的系統，如圖2所示。

該圖顯示微核心系統，它是由同中心的層包圍一中央核心所構成，多少有些類似單體系統，中央核心構成作業

五、發明說明(4)

系統的"核心"。

核心單元MK在此縮減成"微核心"，它僅能執行系統之操作所需的基本機能，即：

- 記憶體管理，
- 5 • 程序管理，至少關於所謂的低階程序--與工作的切換有關。此指出，某些微核心系統在伺服器的協助下管理高階的程序，然而，其它的一些則是由微核心本身管理，
- 埠(信箱)及訊息之傳送的管理，
- 10 • 在各不同驅動程式間仲裁硬體資源的配置(驅動程式是一程式，它控制擴充卡或任何周邊的操作)。

傳統上由單體系統之核心執行的其它機能現由伺服器S完成。這些伺服器構成上層應用軟體A與程式庫之L層間的中間層，L層直接包圍微核心MK。它們的優先權高於應用軟體A。

15

此種微核心架構使得升級作業系統不必重新編譯整個系統；因此，此第二系列的作業系統容允某些單體系統所沒有的彈性。

不過，此系列的架構仍是圍繞著中央核心構建連續的層，且作業系統中所有內部的通信交換都必須通過微核心。

20

事實上，要通過系統之連續層的通信數量多於單體系統的情況，因此，多出許多額外的處理，這使的系統的整體工作速度下降。

此外，對任何新發展的系統而言，都必然要使這些通

五、發明說明 (5)

信交換最佳化，因此，程式設計師的工作變得複雜。

因此，很明顯，雖然微核心系統比單體系統具有彈性，但操作的處理變得冗長，因此，它只是一不完美的解決方案。

5

➤ 關於現今架構的其它評論

此外，現今的系統(單體及微核心)都是以核心單元為中心，連續的層按預先決定的存取特權層層地包圍著核心。

事實上，在所有這些系統中，每一個程式都是按照系統本身固有之永久不變的類型有系統地與指定的層相關，因此，特權的等級將有系統地伴隨著此程式，此不在程式設計師的控制之下，因此構成現有系統的第一共同缺點。

因此，程式例如是：

- "核心"或"監督程式"或"系統"碼(特權碼，它管理與硬體的互動)，
- 被動機能的"程式庫"單元(數學的、影像解壓縮等)，
- 套裝軟體碼。

這些不同類型的碼通常對應於不同的檔案格式，且是由作業系統之不同的單元做不同的管理。

因此，不可能以單一種工具顯示安裝在系統中所有的程式；同樣地，在系統的操作期間，通常也不可能更換"核心"部分，它構成一不透明且不可分割的區塊。

因此，必須使用不同的工具產生這些不同類型的碼並除錯，因此，在所有情況下的發展作業都很複雜；這構成

五、發明說明(6)

現有系統的第二共同缺點。

此外還須瞭解，現有系統的核心是系統的絕對中心。由於此核心的特權等級，因此它具有一特殊的狀態。它構成系統中最難攜行的部分，且無法很容易地修改。

5 此外，現有軟體系統的另一共同特徵是它們被設計成要執行：

- 功能分析階段，此階段使用所要完成的功能來描述系統，藉以構成一非常直覺且可立即理解的描述層，
- 10 • 接著是寫入及程式的編譯，在此期間，此層的可理解性與可讀性被不利地轉變，程式按照預先決定的操作方式構成不透明的程式集。

這些系統的程式是在"智慧型自動裝置"(automata)的請求下操作，核心僅反應應用軟體所陳述的需要而工作。

15 因此，很明顯，雖然核心可修改資源並能使用最佳化功能以增進系統的整體操作，但它總是在下游反應應用軟體陳述的需要而工作。此構成目前之系統的另一基本限制。

另一方面將如所見，在本發明的情況中，系統中所有組件(component)的操作方式是結合成一體，且在功能分析
20 階段期間建立的功能鏈結被保存在系統的架構中。

因此，相較於已知的系統，如以Java語言(已註冊的商標)為代表，對其的描述見於"Introduction to programming using Java", Version 3.0, summer 2000" by D.J. ECK--Department of Mathematics and computer science,

五、發明說明(7)

Hobart and William Smith College, Geneva, New York, US
24/05/2000, 2001年3月7日可在網際網路上取得，網址為
<http://math.hws.edu/eck/cs124/downloads/Javanote3.pdf>>

，它有利於在建立實體間的關係中提供某種程度的額外彈
5 性。

須瞭解，"實體"一詞在已知的語言中(例如Java語言)
代表物件(object)以及可配置的類別(class)。

事實上，在Java語言中，如果某一實體可以實際地呼
叫其操作所需的其它實體，程式設計師必須明確地指定那
10 一個實體應被載入系統的隨機存取記憶體，以允許指定呼
叫的該實體能正確地操作。

因此，很明顯，此類型熟知的語言確實提供了一些與
物件結構相關的彈性，儘管如此，強迫要在設計時選擇特
定的實體還是大幅限制了系統的彈性。

15 此外，這類已知的系統在載入操作期間不會送回任何
錯誤訊息，因此，有可能將有瑕疵的實體或無能力執行其
角色的實體載入系統。此外，該實體的缺陷也只有在系統
的整體操作期間或在特別安排的測試操作期間才會被注意
到。

20

➤ 發明的目的

本發明的主要目的是提出一種作業系統，該作業系統
能提供彈性以及即使現今最強大之作業系統都無法想像的
執行速度等特性。

五、發明說明 (8)

本發明的另一目的是提出一種軟體系統，該軟體系統能克服上述及所有現行系統相關的缺點。

本發明的另一目的是提出一種軟體的解決方法，適合內建系統之設計中固有的限制。

5 為達成這些目的，本發明提出一種發展電腦計劃 (project) 的方法，且意欲由一電腦系統實施，該方法包括：功能分析階段，其目的是將所要提供的功能歸類 (cataloguing)；在電腦機構的輔助下產生計劃之組件以便完成該些功能的階段；在產生階段期間產生組件之同一系統
10 上宣告 (declaration) 的階段；以及將該些組件載入系統之記憶體區供系統實施的階段，其特徵為：

- 在組件產生階段期間，每一項功能完全是由與其相關的組件完成，
- 在組件宣告階段期間，所產生的每一個組件宣告給
15 系統以載入組件，以及
- 在組件載入階段期間，提供機構結合被載入之組件所請求的每一項功能以自動搜尋提供該功能的組件。

按照本發明之方法較佳但非限制的態樣如下：

- 20
- 執行該自動搜尋同時將條件列入考慮，
 - 該條件是以與每一個組件個別相關之屬性 (property) 的函式所表示之標準的函式表示，
 - 在組件產生期間，執行以下的步驟：
 - 產生組件的介面原始檔，其目的是構成組件的"

五、發明說明(9)

介面"部分，

➤ 產生至少一個可實現組件功能的碼原始檔，其目的是構成組件的"實施"部分，

- 5
- 產生每一個碼原始檔的目的是為構成組件的"實施"部分，它可以使用任何程式語言有系統地闡述(formulated)，
 - 經由配置軟體及/或硬體機構，每一項組件可以完成與其相關的功能，
 - 每一項組件的介面檔中包括對該組件所能執行之功能的描述，與該組件完成此功能的方法無關，
 - 介面檔中也包括組件為完成其本身功能需從其它組件匯入之功能的描述，
 - 每一項組件都能被其它能匯出相同功能的組件動態地更換，
- 10
- 該更換按照以下的步驟配置：
- 儲存要被更換之組件的目前狀態，
 - 載入新組件並初始化，
 - 將要被更換之組件的目前狀態傳送給新組件，
 - 建立呼叫功能之組件與新組件間的通信通道，
 - 卸載被更換的組件，
- 15
- 如果新組件的載入或初始化沒有被正確地執行，提供機構用以返回一錯誤訊息，
 - 在組件的產生期間，也執行以下的步驟：
- 產生製作檔，其中明確說明：
- 20

五、發明說明(10)

- ✓ 在產生碼原始檔的步驟期間，產生原始檔所使用的語言，
 - ✓ 此碼原始檔對應於組件的"實施"部分，
- 5
 - 產生組件的原型，
 - 編輯組件之"介面"部分與"實施"部分間的鏈結，以便產生可執行的組件原型，
 - 配置製作檔以便個別地編譯組件，並為組件之"介面"部分的介面原始檔以及為組件之"實施"部分的每一個碼原始檔構成各自的物件檔(object file)，
- 10
 - 將前一步驟期間所產生的物件檔包封到單一的可執行模組中，
 - 在產生組件的原型(prototype)期間，所產生的原型包括：
 - 15
 - 編譯組件之介面原始檔所產生的物件介面，
 - 關於組件之"實施"部分之每一個碼原始檔的原型物件檔，
 - 在編譯組件的介面原始檔期間，需被執行：
 - 20
 - 分析及確認介面之原始檔的一致性，
 - 將介面的原始檔轉換成二進位資料，
 - 至少自動產生一個檔，以便呼叫組件所要匯入的功能，
 - 在編輯組件之"介面"部分與"實施"部分間之鏈結的步驟以便產生可執行的組件原型之後，執行完整性

五、發明說明(11)

自動測試步驟，在此期間確認計劃是否前後一致且封閉，

- 在完整性自動測試期間，在可存取的組件表中自動搜尋能執行計劃所需之每一項功能的組件，
- 5 • 如果確認有數個組件能執行所需的功能，即可在這些組件中選擇某特定的組件，
- 如果完整性測試顯示出計劃沒有正確地封閉，即發出錯誤訊息以指示前後不一致，以便修改某些組件及/或增加某些組件，
- 10 • 對載入系統的每一個組件而言，對該組件為能執行其本身之功能需要從其它組件匯入之功能的描述永遠可被該自動搜尋組件的機構存取，
- 在將前一步驟期間所產生的物件檔包封到單一的可執行模組期間，確認計劃之各不同介面部分所供應的功能都已在相關的原始檔中被確實完整地描述且已被編譯，
- 15 • 該方法包括將載入系統之組件的映圖(map)儲存到系統的隨機存取記憶體區內，
- 當發出呼叫的組件呼叫一功能時可查閱該映圖，以便決定該映圖的組件是否能供應其所呼叫的功能，
- 20 • 該自動搜尋組件的機構可以查閱該映圖，
- 在系統的操作期間該映圖被即時更新，是被載入系統之組件的函式，
- 每一個組件可與一特權等級相關，它是該組件存取

(請先閱讀背面之注意事項再填寫本頁)

裝
訂
線

五、發明說明 ()

其它組件之可能性的條件，是該其它組件各別特權等級的函式，

- 提供卸載機構，用以自動卸載其所提供之功能不再為電腦計劃之操作需要的每一個組件，
- 5 • 該機構包括每一個組件的使用計數器，
- 該計數器計數匯入與計數器相關之組件之功能的組件數量，且一旦它的使用計數器遞減到零，該組件即被卸載。
- 提供一機構用以自動釋放該被卸載之組件所使用的資源。
- 10

➤ 圖式簡單說明

- 圖1表習用之作業系統之單體結構概圖，
- 圖2表習用之作業系統之微核心結構概圖，
- 15 • 圖3a到3c代表按照本發明之作業系統的3個概圖，
- 圖4代表按照本發明之系統的一個組件的概圖，
- 圖5代表構成此組件之步驟的概圖，
- 圖6說明按照本發明產生及配置作業系統之組件的邏輯圖，
- 20 • 圖7代表按照本發明分配及載入系統之軟體組件的模式，
- 圖8說明按照本發明在動態載入及卸載系統之組件期間減少記憶體之消耗，

五、發明說明(13)

- 圖9說明按照本發明動態更換系統中之組件的邏輯圖，
- 圖10a、10b及10c是載入按照本發明之系統之組件的3個映圖。圖10a對應最小的結構，其中，按照本發明之系統僅只有用於啟始作業所需的組件被載入。圖10b對應一基本結構，它進一步允許多分緒(multi-thread)管理。圖10c對應於按照本發明之系統的增強型。

10 > 系統概述

前已描述，現今的系統，無論是單體或微核心，是按照與它們相關的系統層級有系統地區分程式。

- 按照本發明的系統，不是按此方法區分程式。反之，至於本發明，作業系統是由完全無層級的組件所構成。因此，是按照原本的特性，按照本發明的系統並不顯現層的結構。

- 此外，至於本發明，雖然軟體系統的程式設計師仍要保留與系統中每一組件相關的特定特權等級以便與其它組件通信(此特權等級可能對應於習知系統中所習用的--"核心"、"程式庫"、"套裝應用軟體"，或按請求所產生)，但程式設計師不用為每一個組件產生此種相關，也不用為組件選擇任何所想要的特權等級。

須指出，"程式設計師"指的是個人或發展團隊。將可瞭解，按照本發明之系統的主要益處是允許發展團隊以最

五、發明說明(14)

有效率的方式工作，每一位發展人員都可經由發展一組件負起對應用軟體之功能應負的責任。此外，按照本發明，特權的細部管理是由系統提供，開放極多的可能性，例如：

- 5 • 組件是在"無特權"模式(套裝應用軟體)中產生、發展及完成測試，一旦發展結束且完全確認有效後，則是在"有特權"的模式(系統)中使用。在"系統"模式中，經由設計一應用軟體型式的"驅動程式"，使此方法具有與微核心系統相同的微調彈性，以便隨後在與單體核心同等的性能水準中使用。
- 10 • 在一群類型相同的組件(例如"資料編碼/解碼器")中指派不同特權的可能性：要求速度時，使用系統組件型式的簡單組件(例如：GIF"資料編碼/解碼器")；不需要快速但要求安全並要提供完整之錯誤容
- 15 HTML"資料編碼/解碼器"或Marcomedia Flash(註冊商標))。

此選擇是以逐一的方式進行，允許程式設計師完全按照他的目標建構他的系統。

- 圖3a到3c以概圖說明按照本發明之系統的結構，它完全是以組件的方式實施，且它不包括"中央"單元，如核心或微核心。
- 20

如下所見，這些組件每一個都是以其本身存在於系統中；它們被產生、編譯、載入系統的記憶體中、並分別單獨地被卸載，藉以能：

五、發明說明(15)

- 動態地更新系統(新版本，修改錯誤等)不需要重新啟動，
- 經由提供按終端產品特有的需求改變硬體管理策略的可能性，以避免設計軟體系統期間發生硬體無法被軟體系統移動。

5

按照本發明之系統中的每一個組件只專注一項功能，這與習用作業系統中的核心不同，如同絕大多數的程式，習用作業系統中的核心要執行多項功能。

- 此外，為完成與其相關的功能，每一個組件可以請求一或多個其它組件執行一或多個其它功能。

10

因此，按照本發明之系統的操作，是將每一個指定的組件載入系統的記憶體，與能執行指定組件本身操作所需功能的其它組件通信。

- 因此，圖3a到3c中的箭頭說明系統在某些組件間所建立的通信通道，以便允許它們之間相互傳送功能。

15

這些圖也說明系統之結構的動態特性，如圖3c所示，它的組件31可以由組件32取代以便完成相同功能，且不需要改變任何其它組件。這些態樣將在下文中詳細說明。

- 20 ➤ 組件--一般考慮

■ 一般定義：

須瞭解，按照本發明，組件是系統的基本單元，因此，精確的定義十分重要。

就某種意義來說，組件是孤立、獨立且自主的碼單元。

五、發明說明(16)

如以下更詳細的描述，按照本發明之系統的每一個組件具有以下性質：

- 它專注執行一獨有的功能，
- 它完整地執行此項功能。某項功能在單一個組件內執行。此使得在執行一應用軟體或任何其它程式時
5 如果想要改變某功能，所需修改或更換的組件不超過一個，
- 正如物件的哲學，其方式是完成某項功能的組件，並不知道使用此功能的其它組件，
- 10 • 每一個組件都是被單獨地編譯，以便構成獨立的可執行檔，它可鏈結到組件英(稱為組件的"模組")。在編譯期間不編輯組件間的鏈結。此允許前所未見的彈性，特別是對於執行系統之基本功能且在傳統系統中被配置在"核心"中的組件，
- 15 • 每一個組件在介面的協助下與包封在它模組中的可執行碼相關，同樣地，介面也個別地與組件相關。在設計組件期間，程式設計師在以應用軟體特別為此目的設計的描述語言的協助下，以本文檔描述組件所供應的功能，本文檔構成組件之介面的原始型式。此描述是系統所能獲得有關於組件的唯一
20 資訊。此包封保證不會出現"邊緣效應"，也不會出現系統之組件間無法控制的相互依存關係。

因此，按照本發明，系統的組件是一個有介面協助之完整包封(在模組中)的軟體物件，與系統中的其餘部分隔

五、發明說明(17)

離，它具有以下的性質：

- 重複使用性及模組化性(一個及相同的組件可以配置在不同的應用軟體中--或更廣義地說是在各不同的程式中)，
- 5 • 環境的獨立性(每一個組件都是個別地產生，能單獨地載入系統或從系統中卸載)，
- 動態鏈結(組件是根據請求插入而非有系統地插入)，
- 以及，專門限定於某一指定功能。

10 組件C由兩個主要部分構成，如圖4所示：

- IMP部分，它用於實現與組件相關的功能，習慣上此部分稱為組件的"實施"部分。此部分對應的碼用以執行以完成組件的功能，及/或在硬體單元上實施的硬體。須瞭解，"碼"指的是一或多個程式，就
- 15 其本身而論，它可以由任何已知的語言撰寫而成，該些程式意欲完成與組件相關的功能，並為此目的執行一或多項功能，
- 組件的"介面"部分INT，它是描述部分，允許與系統中的其它組件交談。

20

■ 組件的包封

每一個組件被包封在"模組"中，一可執行檔鏈結組件之介面與碼所對應的物件檔。將如以下所見，此檔是以自動的方式為每一個組件產生，按照本發明，與構成系統的

五、發明說明 (18)

其它組件無關。

為產生組件，需要定義位在組件外的功能，以及在執行它們期間所需使用的結果；此來自習用的功能分析設計。

另一方面，如所瞭解，按照本發明，系統將保存此功能分析階段所產生的語義資訊，一直保存到組件開始執行。

此極有利於系統的升級及增加它的可讀取性。

在應用軟體所產生之從屬語言(dependencies language)的協助下，組件外部的功能描述於組件的介面原始檔，在前文中已提及。

更明確地說，與組件C相關的每一個介面原始檔INT可以完全描述此組件。它描述那一個介面單元必須從其它組件匯入，以及，那些匯出到其它組件。

此檔將被編譯(在編譯檔MF(也稱為製作檔(makefile)的協助下)並與對應於"實施"部分IMP(亦即組件之"實施"部分的每一個原始檔)之碼的物件檔鏈結，以構成組件的模組M，它構成組件的可執行型式，如下文中的圖5所概示。用來產生介面原始檔的從屬語言將於本文稍後解釋。

接著，將模組M載入目標50(目標的定義是供按照本發明之軟體系統操作的電腦硬體)，並立即被此目標上的其它組件識別，如以下的詳細說明。

■ 組件的實施部分

實施部分IMP是組件的操作核心。此部分可以有效地實現組件的功能。根據碼所配置的功能，實施部分可以各

五、發明說明（19）

種方式實現功能。

因此，不同的組件每一個可以實現一種且相同的功能，不過，同時考慮不同的標準(criteria)，該標準可以在設計包括該組件之應用軟體或程式期間預先決定，或在實
5 際執行應用軟體期間動態地決定。

這些標準是以屬性(property)的型式描述(例如空白(void)、字元(char)等)，這些屬性與組件的介面相關，將在本文稍後描述。

標準是以這些屬性其中之一的函式來陳述，它是定義
10 在匯出功能之組件的介面內，匯入功能之組件的介面將此屬性與一比較運算子(<、>、=等，如下文中的描述)以及比較值聯結。

此外，可以按組件(例如是硬體)外部的標準，或按系統目前的狀態，或在應用軟體執行期間靜態或動態地修改
15 系統或應用軟體。

因此，每一個組件的介面檔供應一組變換資訊(meta-information)，它允許系統的特定組件(從屬管理員)以一相關的方法，按照先前提及的標準管理組件間的關係，其標準與組件的型式、此組件的位置、系統之狀態的
20 資料項等有關。

還有一點須指出，如果每一項功能可由碼唯一地實現(以"軟體"實施功能)，它也可能由至少一個硬體單元完成(以"硬體"實施功能)。

五、發明說明(20)

■ 組件的介面部分

介面部分可精確地描述在組件執行期間所能供應的功能，以及必須供應何種功能給它。

它以應用軟體發展的從屬語言定義要供應給組件做為
5 輸入資料的格式，以及組件輸出資料的格式。此格式只與所要實現的功能有關，與實施部分實現功能的方法無關。

此介面部分描述組件所能實現的功能，與實施部分實現此功能的方法無關。事實上，無論碼(或經規劃的硬體單元)以何種模式配置功能，此功能仍保持相同。它執行相同
10 的處理操作，在輸入請求相同的資訊，並在輸出供應相同的資料。

因此，介面部分可以使系統充分地升級。因此，由軟體實施的功能(對應於一組件，它的實施部分是由碼構成)稍後可由硬體實施實現。

15 在此情況，對應的組件包括一介面檔，但它的實施部分是由非常簡單的程式構成，描述與硬體單元的鏈結。

反之，硬體實施可被軟體實施的功能取代，不會對其它組件有任何衝擊。

在組件之模組的描述中，介面是以關鍵字"介面"定義。

20 以下的表描述功能的各種不同屬性；這些屬性與組件請求的功能相關(方向=匯入)或與組件供應的功能相關(方向=匯出)。

五、發明說明(21)

屬性	方向	描述
唯一(unique)	匯出	在本地的系統上可存在此介面的單次實施。因此，系統僅載入第一個實施，且拒絕後續的實施。
外來(foreign)	匯入	從另一個組件匯入此介面。
載入時間(loadtime)	匯入	此介面與使用它的組件同一時間被自動地載入。
執行時間(runtime)	匯入	介面在使用它之組件的請求下載入(動態管理--見有關於從屬管理員一節)
需求的(required)	匯入	介面是操作時不可或缺的。 如果與此介面相關的功能未被實際地匯入，組件將不會被載入。
多樣的(multiple)	匯入	匯入所有能匯出此介面所請求之功能的組件。

在各種不同屬性的協助下，介面可以定義在它的本文檔內。例如：

```

5      inprocess supervisor module {
          interface component {
              /*Definition of the exported interface*/
          }
          foreign loadtime required interface codec {
              /*Definition of the imported interface*/
10      }
      };

```

⇒ 介面的屬性

介面的屬性使其可以定義本文稍早前所提及的標準。

15 它們是以C語言(註冊商標)定類型。下表描述屬性的各種類型：

五、發明說明(22)

屬性	描述
空白(void)	空白(與類型C相同)
字元(char)	8-位元整數
短(short)	16-位元整數
int, long	32-位元整數
長長(long long)	64-位元整數
浮動(float)	浮動
加倍(double)	加倍(與類型C相同)
struct name	結構
union name	結構
字串(string)	UTF-8字串，以零結束
布林(Bollean)	布林
memarea	記憶體區
enum name	計數

以下是與這些屬性相關的延伸：

延伸	描述
有符號(signed)	有符號的值(原設值)
無符號(unsigned)	無符號的值
const	常數值

- 在描述介面檔中的從屬關係期間，運算子使得它可指
 5 定介面的屬性。因此，唯有另一介面的屬性值對應一定義
 明確的標準，它們才可能認可該另一介面匯入。

以下是與介面之屬性相關的運算子：

運算子	描述
in	此參數是輸入(原設值)
out	此參數是輸出，且是由被呼叫的功能提供

五、發明說明(23)

inout	此參數是輸入/輸出
=	指派的運算子
==	同C語言
<	同C語言
<=	同C語言
>	同C語言
>=	同C語言
!=	同C語言
&&	布林AND 邏輯AND的結果不為零
!&	(&&)的反相 邏輯AND的結果為零

- 關鍵字"undef"可做為屬性或功能的比較值。在以下的例中，假設組件需要一名稱為codec的介面，它包含一名稱為mainType之32位元的整數屬性，它的值必須剛好為2。此
- 5 介面必須確保功能Func{}取得一字串做為參數，並返回一個32位元有符號的整數。

```

foreign loadtime required interface codec {
    /* Definition of the imported interface */
10    int mainType == 2 ;
    int Func(string name) != undef ;
    ...
}

```

- 15 因此，功能的供應與請求是經由規劃介面供應或請求的描述完成。例如組件可以供應"排程員"(scheduler)規劃介面(排程員可以產生/消滅/操縱程序及分緒及它們的同

五、發明說明(24)

步)，同時請求"記憶體"介面(管理記憶體)、"計時器"介面(管理排程員所需的計時器，以在指定的時間之後中斷程序，並前進到下一個程序)等。

這些規劃介面可包含：

- 5 • 屬性(型式、記憶體管理的類型、序列的屬性等)；
- 以及為執行組件所呼叫的方法(函式)。

屬性可用於某介面的供應期間與請求期間。因此，此可能表示的事實是，與組件相關之記憶體介面的型式是1.5.2(在介面的供應期間表示的屬性)；否則，事實是記憶體介面請求的型式至少是1.0.0，但不是1.0.4，因為它知道
10 具有無法避免的"錯誤"(在介面請求期間表示的屬性)。

此外，介面的內容(屬性與方法)構成一描述，是由程式設計師在以此為目的之應用軟體所產生之語言的協助下所撰寫。

- 15 此語言是由從屬編譯器以二進位的格式編譯，它包括在由模組所構成的可執行組件中，與組件的碼及相關資料的能力相同。

此二進位的格式很容易供系統根據所請求的功能執行，以便交互鏈結組件。

- 20 此語言也用來指定組件所包含之碼的類型("系統"、或是"被動"(程式庫)或"主動"(應用軟體)等)，以及何種類型的碼可以存取某指定功能(例如以系統的"危險"功能限制"系統"的呼叫者)；如此可以選擇最佳的通信方式。

五、發明說明 (25)

➤ 組件--產生及載入系統

現請參閱圖6，以下將描述按照本發明產生系統之組件的步驟，以及在系統中按照本發明的實施。

圖6是一邏輯圖，其中，行代表按照本發明用來產生及配置系統之組件的各不同單元。從左到右，此表的構成包括：

- 兩行"COMP"對應於組件本身，最左行的"INT"對應於介面部分，次一行的"IMP"對應於實施部分，
- "MF"行用於製作檔(makefile)，它是組件編譯檔，
- 10 • "COMPIL DEP"行對應於組件設計單元，它是從屬編譯器，是應用軟體為編譯組件所特別發展的單元，
- "COMPIL CODE"行對應於檔案之已知類型的編譯器，檔案構成組件的碼，
- 15 • "LINK"行對應於鏈結編輯器，
- "KBIN"行對應於按照本發明之系統的特定發展工具，在計劃的設計期間進入(它的概念將在後文中定義)，
- "GD"行專用於按照本發明之系統的特定組件，稱為從屬管理員(或GD)，它構成按照本發明之系統的核心，但與習知技術的核心或微核心完全不同，
- 20 • "LOAD"行對應於按照本發明之系統的特定組件--載入器，它可將系統的組件載入目標的隨機存取記憶體區，

五、發明說明(26)

- 最右行為所描述之各階段的指示。

■ 組件的產生

- 在此定義"計劃"(project)的一般概念，按照本發明，它
- 5 在系統中指定一組具有共同目標的組件。因此，計劃對應於一應用軟體，但更廣義地說，它對應於可由電腦系統執行之任何類型但不同於應用軟體的程式(例如周邊管理員)。

- 在產生欲在按照本發明之系統中實施的計劃之前，要
- 10 先按習用的方法在601執行功能分析(AF)階段。

在此階段定義所需的各種組件，它們各自的功能，以及與這些功能每一個相關的需求(必須有其它組件為其產生的功能)。如以下所見，可以使用已載入系統的組件提供計劃某些功能。

- 15 此預先的分析一旦執行，且也瞭解了組件的需求，程式設計師可以為每一個組件繼續執行步驟602到606：

- 在步驟602，使用應用軟體所發展的從屬語言產生組件的介面原始檔，其中描述組件的特徵。此語言的特徵及它的主要的可能性將於稍後返回此本文，
- 20 • 在步驟603，程式設計師接著產生碼，亦即原始檔，使組件能以一或多種已知類型的語言提供功能。在此步驟期間，組件所要匯出的函式(function)僅是以原型定義，對應的碼尚未撰寫。

特別指出，當啟動程序進行時，要從事與功能對應

五、發明說明(27)

5 的處理或操作(對功能的描述將以法文陳述)，例如對記憶體的管理，以便管理它們對目標之處理器、圖形顯示器等之取存。吾人所謂的函式是一軟體碼部分，它也從事處理操作。函式是轉換成功能之程式語言的軟體(因此得名)。

在所有的程式語言中，所採行的函式與吾人的系統非常類似。它包括：

- ✓ 原型，它定義參數的輸入與輸出。此原型可以鏈結到吾人之組件的介面部分(它的角色相同)，
- 10 ✓ 碼部分。此部分執行資料處理，因此，它可以決定供應到輸出的結果。

還須指出，函式的原型定義函式的名稱，它的輸入參數(它們的名稱及它們的類型)，以及函式所返回之結果之資訊的類型。

- 15 • 在步驟604，程式設計師產生一製作檔，他在其中特別指定：
 - ✓ 在前一步驟603期間產生原始檔所使用的語言，
 - ✓ 與組件之實施部分對應的原始檔，
 - ✓ 編譯選擇，
- 20 • 在步驟605，程式設計師使用從屬編譯器以及與組件之原型所使用之語言相符的編譯器產生：
 - ✓ 物件介面，以從屬編譯器編譯在步驟602所產生之組件的介面原始檔所產生。為編譯此介面，從屬編譯器分析介面之原始檔的本文，確認它的一

五、發明說明(28)

致性，接著將其轉換成DG所能瞭解的二進位資料。從屬編譯器自動產生檔案用以呼叫介面中所列舉之要供應給組件的功能。

5 這些檔是從屬編譯器所產生之"包括"(include)類型的檔。"包括"檔是為每一個碼實施語言所產生。這些檔的角色是在電腦系統中定義一個命名空間(易言之，有一對應的前後文與其相關)，它在組件的外部，且與其不同，命名空間使得它可以存取組件在它介面中所請求的功能。因此，這些"包括"檔每一個定義的結構都可以被組件的
10 介面呼叫，按此方式，介面中所做的請求轉變成可供與"包括"檔相關之實施語言使用的結構與名稱，

✓ 為步驟603所產生的每一個原始檔產生物件檔；
15 在此連接點，每一個物件檔對應一個原型原始檔，它的功能沒有被撰寫完全，

- 在步驟606，程式設計師接著編輯物件介面與組件之實施部分之物件檔間的鏈結，按此方式，產生一可執行的組件原型。

20 接著，在步驟607，KBIN對為擬想之計劃所發展的所有原型組件進行完整性測試，確認所產生的計劃是前後一致且封閉的計劃(易言之，計劃所需要的所有功能都出現在此計劃中)。

在完整性測試期間，提供表單及對所有組件之介面的

五、發明說明(29)

描述給KBIN。因此，KBIN可以在這些組件間搜尋能完成計劃所需功能的組件。如果有數個組件都能完成所請求的功能，KBIN提示程式設計師在這些組件間做選擇。接著，由KBIN自動尋獲或是由程式設計師指定的組件加入到構成計劃的組件中。KBIN也要嘗試解決此新組件所需要的其它功能。因此，KBIN使得可以自動地解決計劃所需要的絕大部分功能。

如果計劃沒有正確地封閉，KBIN發出一錯誤訊息給程式設計師指出不一致性，後者將重新執行連續的步驟602到606，以便修改及/或增加應用軟體中的某些組件；如果計劃前後一致且封閉，經由撰寫組件之實施部分之原始檔中完整型式的函式，程式設計師在步驟608完成此實施部分之碼之函式的原型。

在步驟609，程式設計師配置製作檔以各別地編譯每一個組件，按此方式產生碼以使物件檔能執行組件的功能。在此步驟期間，將製作檔帶入從屬編譯器及用來產生碼之語言的編譯器。為目前正在發展之計劃中的每一個組件重複步驟608及609。

在步驟610，程式設計師接著配置鏈結編輯器以便為每一個組件產生可執行的模組，該模組與在編譯步驟605及609期間所產生的物件檔組相關(在步驟605期間所產生的是物件介面，然而，與碼相關的物件檔則是在步驟609期間產生)。

在編輯此鏈結的期間，它對應地將組件包封到模組

五、發明說明(30)

內，編輯器確認計劃之各不同介面所供應的所有功能都在相關的原始檔中被確實地完整描述，並已經過編譯。

■ 組件的宣告

5 一旦為計劃的每一個組件產生了可執行的模組，程式設計師必須宣告這些組件的存在，以便它們可被系統納入考慮。此宣告是在步驟611執行，在此步驟期間，程式設計師將每一個組件宣告給"載入器"，載入器它們本身是按照本發明之系統特定的組件。

10 在步驟611期間，被宣告了組件的載入器將組件儲存到載入器之個別的表中，載入器將宣告到該處的組件組聚集在一起。

就整體而言，此步驟611也是與目前所發展之計劃在某種程度上相關的步驟，關於計劃所發展的組件組被宣告給
15 至少一個載入器--每一個載入器對應一個電腦系統的媒體(碟片、網路等)，與此載入器相關的組件儲存在其上。

■ 將組件載入系統

以下將描述產生與宣告包括數個組件之"計劃"的步
20 驟。現將在下文中描述這些組件的配置，此部分說明於圖6的下半部分。

在步驟620組件C1呼叫它所需要的功能F1。

此呼叫將發生兩種可能：

- 在組件C1的介面中，功能F1的屬性被定義成"載入

五、發明說明 (31)

時間"("載入時間"這個屬性已在前文中描述)。若為此情況，在載入組件C1之前，GD將搜尋能完成此功能F1的組件並將其載入系統。

5 ✓ 如果此組件本身需要"載入時間"這個功能，在載入組件C1之前，GD還是要先載入能完成此項功能所需的組件。因此，DG將遞歸地載入它所遇到能完成各種不同"載入時間"功能所需的所有組件(如何將組件載入系統將在下文中進一步解釋)，

10 • 或者，功能F1的屬性被定義成"執行時間"("執行時間"這個屬性已在前文中描述)。接著，組件C1必須明確地呼叫由GD供應的另一個功能"depmgr"以呼叫"執行時間"功能F1。為做到此，在C1的介面中，"depmgr"功能絕對必須隨"載入時間"一同宣告，以便一旦C1被載入及初始化(見前文中"載入時間"功能的處理)即能使用此介面。

15

接著，C1在正好要使用它的時刻在步驟620呼叫，以便匯入"執行時間"功能F1。接著，GD將能完成此功能F1的組件載入。如果此組件在被載入的時刻需要一功能(在它的介面中宣告的功能為"載入時間")，則完成此"載入時間"功能的組件也要被載入。因此，GD將迴遞地載入它所遇到能完成各種不同"載入時間"功能所需的所有組件(如何將組件載入系統將在下文中進一步解釋)。因此，唯有將所遇

20

五、發明說明(32)

到的所有"載入時間"功能都解決之後，C1才能使用最初在步驟620請求的功能F1。如果C1在載入時請求的功能有一個沒解決(功能宣告為"載入時間")，C1就不會被載入。事實上，C1將無法完成它的功能任務，因為它無法使用它所需要的另一項功能。

5 如果C1所屬且宣告給系統其中一個載入器的計劃中沒有一個組件能完成C1所需匯入的功能，系統將在另一個計劃中識別可以供應C1此項功能的組件。

基於這些目的，GD(它永遠被載入系統的隨機存取記憶體)經由它的"鏈結圖"搜尋鏈結圖中是否有一個組件的介面物件檔指示此組件能供應C1所請求的功能。

為做到此，GD解釋"二進位"型式的介面檔，該型式是儲存在可執行的組件中，並由"載入器"組件載入/傳送給執行中的GD。

15 GD首先搜尋與所請求(匯入)之名稱相同的介面。當發現相同者，它接著確認所請求之功能的所有屬性是否都出現於其中，以及是否滿足任何限制及條件(版本高於0.6.2等)。

GD鏈結圖是被載入系統之組件的目錄，易言之，它不僅如前文在步驟611中所描述的被宣告，且它的物件介面檔也匯入目標的隨機存取記憶體區中，且它與鏈結圖的其它組件連接，以交換供應及/或請求功能。

鏈結圖是儲存在目標的隨機存取記憶體區中，因此，鏈結圖是被載入之組件的即時影像，此鏈結圖可以動態地

五、發明說明(33)

調整。鏈結圖是唯一的，且與GD相關，GD可存取此鏈結圖的資訊。

在步驟621期間，GD決定C1所呼叫的功能是由鏈結圖中的其中一個組件供應(將該組件稱為C2"被呼叫組件"，同時，C1是"呼叫組件")，它建立此組件C2與組件C1間的通信通道，在627將組件C2的特性匯入組件C1。

更明確地說，GD為C1建構一稱為"視野"(view)的資料結構，它提供關於所要匯入的視野。此"視野"包含屬性的值以及函式(或使得可間接到達它們的碼，例如系統的呼叫、分緒的遷移等)匯入的指標器。此"視野"從"包括"檔中之從屬編譯器所產生的單元直接發出。

如果GD在621決定C1所呼叫的功能不是由鏈結圖中的任何組件供應，它將逐一地詢問系統之載入器的表，以便決定宣告到這些載入器中的組件中是否有能提供C1所請求的功能。對載入器的詢問以步驟622表示。

在此詢問期間GD使用"載入器"功能，此功能具有列出組件供每一個"載入器"觀看所需的函式。"載入器"功能允許GD比較載入器之組件所匯出的功能與需要匯入的功能。它也允許GD從與載入器相關的媒體中將各個組件載入目標的隨機存取記憶體或從其卸載。

要特別指出，按照本發明之系統的每一個載入器與一特定的媒體相關，使該媒體可以儲存組件；因此，每一個載入器可與一記憶體碟片或任何其它可支援記錄的媒體相關，例如CD-ROM類型的碟片。現將特別參考圖7說明複數

五、發明說明(34)

個載入器。

如果有一個這類組件位在系統其中一個載入器的表中(步驟623)，表中包括此組件的第一載入器在步驟625將被識別之組件C2的物件介面傳送到從屬管理員，且GD本身
5 在步驟626將此組件C2的"視野"傳送給組件C1(步驟627)。

C2的"視野"被GD儲存到由DG動態產生的系統記憶體區中，組件C2一旦被載入，在此組件被卸載的同時亦被摧毀。此區域與組件可存取的位址空間相同。

在步驟623期間，如果GD為確認其中一個載入器的表
10 中有一個組件能供應請求之功能的搜尋結果是否定的，系統在步驟624結束。

無論如何，將組件C1載入系統的隨機存取記憶體將引起GD自動搜尋能提供組件C1操作所需功能F1的組件C2，如前文中的解釋。

15 此外，如果組件C1所呼叫的不只是一個而是數個功能，GD也同樣為組件執行自動搜尋以提供所請求的這些功能。

須注意，與習知的系統相較，按照本發明的系統能提供額外的彈性。

20 事實上，已知的系統是以實體建構，程式設計師必須在系統中為每一個呼叫的實體具體指定該呼叫之實體操作所需其它實體的表，且必須載入到系統中以使呼叫的實體能正確地操作。

因此，習知系統的情況是，實體的需要無法以功能表

五、發明說明 (35)

示，必須以特定的實體表示(例如Java語言的情況)。

此外，本發明的情況是，數個不同但能提供相同功能的組件每一個都易於與呼叫的組件C1聯結。

- 此外，本發明其它的優點來自於與組件相關的標準，
- 5 在GD自動搜尋能提供第一組件所請求之功能的組件期間，這些標準被列入考慮。

事實上，在GD搜尋組件的期間，GD不僅搜尋能提供所請求之功能的組件，且能根據本文早先提及的標準，更精確地搜尋能提供該功能且結合有特定條件的組件。

- 10 以現有的語言(例如Java語言--註冊商標)為例，事實上，程式設計師絕對必須在呼叫的實體與被呼叫實體間產生明確的鏈結。因此，很明顯，這類習知系統所提供的彈性，無論如何都無法與本發明相比，因此，本發明能給予其本身更佳的維護與升級(例如，為產生更強的版本)，提供
- 15 所請求之功能的組件，可以由以不同方法但能完成相同功能的組件取代，或者，可以增加能完成該功能的新組件，以便提供GD新的選擇。

因此，本發明可以建構出極具升級能力的系統，其中，可以隨心所欲地更換一或多個想要更換的組件。

- 20 每一個用以更換的組件所提供的功能已由被更換的舊組件提供。

不過，完成此功能的特定機構，它們是軟體及/或硬體，可以任意地適合。

在應用軟體的架構內，諸如安裝在各不同站台之系統

五、發明說明 (36)

之網路的維護，或是裝置之行動單元的維護(例如可攜式電話或個人筆記簿等)，使用者不希望實際回到中央站台才能升級，這可能是最有利之處。

5 因此，安裝在上述類型之裝置中的應用軟體，可以經由從中央站台將一或多新組件下載到這些裝置中，即可很容易地升級，按此方式，不需要實際地移動裝置，該裝置可以散布在任何地區。

此外，更換這些組件，也不需要重新編譯配置它們的電腦系統。

10 如果，提供C1所請求之功能的組件C2已被確認(可能具有一標準能滿足(1)之介面所請求的條件)，且C2的特性在步驟627被匯入C1，依照組件C2是或不是構成鏈結圖中的一員(步驟628)，從屬管理員的操作將有所不同。

15 如果C2已在鏈結圖中，GD直接前進到步驟633以產生C1與C2間的通信通道。為做到此，系統將使用以下所描述的4種通信模式其中之一，或是使用按照C1及/或C2之介面中明確指定的特性所特別產生的特定模式通信：

⇒ 直接呼叫：

20 此為最快速的方法。所請求的功能或方法被直接呼叫，且它的結果也是直接返回。其中沒有中間的碼。當兩個組件都在相同的機器、相同的位址空間及相同的特權等級時，從屬管理員選擇此機構。

位址空間被定義成邏輯記憶體區域，其中，碼的位址或資料的位址對所有存取此區域的組件都具有一義意。指

五、發明說明 (37)

向碼或指向資料的指標器可以從一組件傳送到另一組件，不修改它的值將可以存取相同的資料。

在本發明中，"位址空間"也暗示共同的特權，位址空間定義一"空間"或分類在同一特權等級的物件：因此，指向一個或相同位址空間的物件可被不同的組件以相同的方式存取。

此外，例如也可以定義一個特權等級較低的"使用者"空間，以及具有較高等級的"監督程式"空間。

每一個程序之"使用者"空間與"監督程式"(系統)空間之間指標器的一致性是依從的(電腦系統的處理器同時存取兩個空間)，但"使用者"碼不能參考位於"監督程式"空間中的資料。

特權是組件間存取的可能性；在原始介面中，可能使用以下的特權等級：

15

等級	描述
內部處理 (Inprocess)	模組是在與呼叫方相同的位址空間中執行。
子處理過程 (spawn)	組件在它載入期間所產生之其本身的程序中執行。須指出，當標示有"spawn"的組件被載入時，GD自動請求一"排程員"組件專門用於排程以產生一由C1開始的程序以及對應的位址空間。接著，組件被包含到此程序的位址空間內，並當作主動單元操作(所執行的處理操作可能除了組件的開始之外沒有任何動作)，且在組件沒有"spawn"標示的情況時它不是被動單元(它只是在等待呼叫以便執行處理操作)。GD在步驟621之前區分組件的此特權等級。

五、發明說明(38)

IO	模組執行輸入/輸出操作(驅動器)。
監督程式 (Supervisor)	組件必須在監督模式下執行(最高等級的特權)。

為指定組件為這些特權等級其中之一，程式設計師將組件之模組之定義中的對應參數固定，該定義包括在類型的本文中：

```

5          inprocess supervisor module {
              /*Definition of the module*/
          };

```

因此，一般言之，每一個組件都可個別結合一個指定的特權等級，該特權等級做為此組件存取其它組件之可能性的條件，是該其它組件各別之特權等級的函式。

另一說法是，與組件相關的特權等級[空白](lacuna)給予此組件"觀看"某一組件之介面所提供之功能的可能性，是與這些其它組件實際相關之特權等級的函式。

15 此外，如下文中所見，組件間由與各不同組件個別結合之特權等級所建立的此種存取階層，可以被運算子個別地結合某特定組件加以"短路"。

此種呼叫的例子如下：

- 監督程式組件對監督程式組件，
- 20 • 使用者組件(易言之，從特權等級較低者)對同位址空間中的使用者組件(此種應用軟體使用組件管理共享及被動函式的程式庫)。

此方法只有在記憶體組件沒有提供保護時使用，此可

五、發明說明(39)

供為其產品建構系統的工程師選擇，藉以在沒有保護之下提供極快速的系統。

- 當此組件只提供單一個"系統"類型的位址空間時記憶體組件不"提供"保護。此類型的記憶體管理不提供保護(所有的操作都在相同的空間內，且有可能破壞或修改屬於其它組件的碼或資料)，但這只有在不具有任何"記憶體管理單元"(MMU)的某些目標上才有可能。

⇒ 系統呼叫：

- 10 請求被轉向系統呼叫。後者將參數從使用者堆疊轉移到監督程式堆疊，改變特權等級，完成呼叫，返回較低的特權等級，且結果是直接返回。

當使用者組件(無特權)呼叫監督程式組件的功能時選擇此方法，在此情況時，此功能授權無特權的組件使用它。

- 15 此授權是在從屬描述語言的層次經由關鍵字"提升"(promote)完成。此外，更廣義地說，以下的關鍵字可與組件相關(在以應用軟體發展之語言的語法中，關鍵字只是單純地在屬性或方法前加字首，這些關鍵字改變它們的操作方式)：

20

關鍵字	描述
提升(promote)	使得可以從使用者組件呼叫在"監督程式"組件中實施的函式。
不信任(untrusted)	與提升相同，但不會自動切換監督程式模式。

五、發明說明(40)

非同步(async)	立刻傳回給呼叫方，且該功能繼續平行處理：在此情況，間接呼叫經由"觀看"產生分緒之GD(其本身呼叫在考慮中的函式)之結尾處的碼並立刻交出(在產生的分緒中繼續執行)。
連續的(serial)	與功能同時發生的呼叫被禁止(呼叫的另一方將被置於保留)。

因此，前文所提及的授權，為個別地授權一個組件存取另一個具有無法存取之特權等級的組件，經由配置關鍵字"提升"，可以"短路"已建立的特權等級。

5 ⇒ 分緒(threads)的遷移：

按照本發明的系統，執行分緒的邏輯管理(識別器、優先權、系統堆疊等)與它的實體管理(處理器的前後關係(processor context)、使用者堆疊、其中包含分緒的位址空間等)相互獨立。

10 當一組件呼叫位於另一位址空間中的函式且是同步呼叫時(呼叫方被凍結等候結果)，呼叫方"遷移"到被呼叫方並執行該處理。

為達此目的，分緒的邏輯部分"堆疊"在它目前實體部分的上方，並使其本身連接到在被呼叫方之位址空間中產生的新實體部分。

15 接著，所要功能的碼在被呼叫方中(易言之，在相同的位址空間與相同的包含程序)但是在呼叫方的邏輯分緒中執行。一旦函式結束，該呼叫分緒從它原來的實體部分"去堆疊"，並再度"遷移"到呼叫方中。

五、發明說明(41)

分緒的"遷移"碼也負責在兩個方向傳送參數，複製或再映射記憶體區，字串等，當成參數供應。

此機制避免了昂貴許多的傳統方法，傳統方法包括了訊息、將訊息傳送給被呼叫之組件中的分緒，暫停呼叫方的分緒、分解被呼叫方中的訊息、執行所要的功能、重組與結果有關的訊息、喚醒呼叫方、再度暫停被呼叫方。

這也可避免與同步傳遞訊息相關的諸多問題，如優先權反轉(在此，被呼叫的功能是以呼叫方的優先權執行)，資源的計算(呼叫方的機器時間被倒數計時(counted down))，以及，被呼叫的組件中需要保持一個分緒的"池"(pool)，被呼叫的組件進入睡眠狀態直到接收到呼叫，非必要地佔用了記憶體的資源，接著的是大量的呼叫會使得數量不足(在此，呼叫方"遷移"到被呼叫方內，需要的分緒總是有一確切的數量)。

15 此方法使得位址空間外部呼叫這種最常見的情況大幅地最佳化：在同一機器上同步呼叫。

⇒ 訊息的傳遞：

在以微核心為基礎的系統中，此方法是被有系統地使用，本發明與其的區別是它僅只用於不是來自上述種類的
20 呼叫，諸如：

- 不同步的本地呼叫(呼叫方不停留在凍結狀態等待結果)，
- 遠方呼叫另一機器。

訊息的傳遞在於編碼訊息中所供應的參數，將訊息路

五、發明說明(42)

由到目的組件，解碼訊息，執行函式，儲存訊息中的結果，返回呼叫方。前文所述的此方法特別適合不同步呼叫(結果訊息稍後傳送)，以及遠方呼叫，需要任何方式的網路封包結構。

- 5 現請回頭參閱圖6的描述，通信通道的產生，與呼叫將呼叫組件C1記錄到被呼叫組件C2之實施部分中的函式相關。

除了組件它們自己特定的功能之外，稱為Register()的函式是構成"組件"功能的一部分，它是由系統所有組件供應。"組件"功能包括4個函式：

- 初始化，
- 去初始化，
- 記錄，
- 去記錄。

- 15 現在，如果GD在步驟628決定組件C2不是構成鏈結圖的一部分，它在步驟629呼叫組件C2的載入器，以便將C2帶入目標的隨機存取記憶體，在步驟630呼叫初始化C2的函式，並在步驟631更新鏈結圖。

因此，圖7說明鏈結圖G，GD可將一組件或一組組件
20 P2從第一載入器的記憶體位置72匯入其中，以便允許已出現在鏈結圖中的組件C1(鏈結圖中具有先前從其它載入器之記憶體區71匯入的組件，這些組件是構成計畫P1的一部分)與被呼叫的組件C2連接，組件C2已在步驟623期間被GD識別，步驟623是GD在各載入器間搜尋能提供C1所需求

五、發明說明(43)

的功能。

吾人回憶，事實上，KBIN在步驟607執行完整性測試期間(見圖6)，KBIM確認組件所呼叫的每一個功能，嚴格地說，並不是由同一計畫的組件提供，更廣義地說，是由

5 載入軟體系統中的組件提供。

再回到圖6，步驟631領先步驟633已經描述過。

此外，雖然已描述了被組件C1呼叫之組件C2的載入，但須瞭解，組件C1它本身也是被另一個組件呼叫，當啟動系統時，需要初始化功能的呼叫。

10 為此目的，當啟動應用軟體時，GD請求"互動"(interaction)功能，並搜尋能供應此功能的組件。GD也能匯入請求此功能的組件。在所有情況，"互動"功能將呼叫另一個組件，它依次呼叫至少一個其它組件。

15 ➤ 從屬管理員(dependencies manager)

如前所述，按照本發明，從屬管理員是一永久載入系統的組件。它可以架構出構成應用軟體之各不同組件間的鏈結；這些鏈結是由組件的介面建立。

20 按照本發明的有利態樣，只有對應用軟體及系統之操作有用的這些組件才會被載入，因此，它能最佳化所需要的資源。此最佳化可經由以下其中一種模式達成：

■ 在執行期間組件自動載入

當一組件被載入以便執行時，從屬管理員會分析陳述於組件介面中的從屬，並自動搜尋以解決陳述於其內所有

五、發明說明(44)

的從屬，易言之，供應組件所需要的所有功能，並進行相同的搜尋以供應每一個組件它所需要的功能，以"封閉"從屬的系統。

接著，GD將組件自動載入鏈結圖，解決屬性為"載入時間"的匯入介面，也解決它們的從屬，並在這些不同的組件間產生通信通道，是它們各自位置的函式。

因此，可以按照前所定義的任何一種通信模式建立通信通道；它是由任何資料傳送路徑(為每一種通信模式所特有)構成，有能力呼叫功能及相關的參數，以及恢復功能所執行之處理所產生的返回值。

通信通道一旦建立，組件間即可直接相互通信。

■ 組件的動態載入與卸載

當匯入的介面具有"執行時間"的屬性時，只有當使用此介面的組件需要與介面相關的功能時，此組件才請求載入實施此介面的組件。因此，此"執行時間"的屬性替代"載入時間"的屬性，已在前文中描述。

此管理使得可以很精確地決定必須載入之介面之實施的屬性，它是動態參數的函式，當載入請求功能的組件時並不知道屬性。

例如，"本文"(text)組件可以匯入一編碼解碼器介面以便對接收到的檔案解壓縮，此"本文"組件可以接收數種類型的檔案。

在接收檔案期間，"本文"組件搜尋適合解壓縮此檔案的編碼解碼器介面，例如是延伸檔名的函式。為做到此，

五、發明說明 (45)

它使用從屬管理員的Query()功能，將搜尋參數傳遞給它。

接著，它可以在Open()功能的協助下命令從屬管理員載入組件。接著，它使用此組件的編碼解碼器介面將該檔案解壓縮。

- 5 當解壓縮結束，"本文"組件可以請求卸載使用過的編碼解碼器組件。

因此，它可以動態地載入組件。此可以允許只有在需要組件執行處理時再將它們載入。當它的處理結束時，接著即被卸載。

- 10 按照另一例，應用軟體可以在它的載入期間顯示一影像，接下來就不再處理此格式的影像，或只有在特殊的情況下才處理。

- 15 因此，此應用軟體可以動態地載入用以處理影像格式的組件，請求它處理檔案以便顯示影像，並接著卸載此組件。只有當在需要時，該組件才被載入記憶體，這使得記憶體資源可供各不同的組件共享，它是在某一瞬間所要進行之處理操作的函式。

因此，此方法可以大幅減少所需要的記憶體資源。

- 20 從屬管理員也可以自動管理組件的載入與卸載，是它們的使用統計與記憶體資源的函式。因此，此管理變得對應用軟體的程式設計師完全透明。

圖8顯示在記憶體資源少的系統上動態管理組件的益處。在此例中，應用軟體顯示GIF及JPEG兩個影像。每一個影像是順序地處理，藉以可以動態地載入與卸載組件。

五、發明說明(46)

在本例中，組件的動態載入與卸載使得所需的最大記憶體資源可以減少。

此外，在絕大部分的情況中，此載入與卸載不會對性能有可察覺的影響。

5 事實上，組件之模組的大小已縮小(從大約數百位元組到數千位元組)。因此，將它們載入記憶體的操作可以非常快速地執行。此外，絕大多數應用軟體是執行順序處理操作，且輸入的資訊通常是由使用的人供應，輸出的資訊也是供應給使用者。

10 因此，以人的尺度而言，將不會察覺載入及卸載組件所需的數微秒；以圖8的說明為例，操作的執行是按以下的順序：

- 載入影像(載入JPEG碼)，
- 在螢幕上顯示此影像，
- 15 • 使用者，
- 載入另一影像(卸載JPEG碼及卸載GIF碼)，
- 顯示此另一影像。

載入與卸載的時間與其它的處理操作相較微不足道。

■ 從屬管理員使資源最佳化

20 從屬管理員為每一個組件管理使用計數器。此計數器計數組件的匯入物數量，易言之，組件從結合有計數器的組件匯入功能。一旦組件的使用計數器被遞減到0，它即為卸載。

當組件不再使用，從屬管理員自動地將此組件使用的

五、發明說明(47)

資源釋放。

須注意，因此，本發明能使系統所配置的資源特別有效地最佳化，不增加系統一般操作方式的負擔。

事實上，習知的系統一般是配置周期性的機構用以確認組件的狀態，該機構是由特定的作業配置，它周期性地"接管"以便識別要被卸載的組件。

此種周期性的確認工作並不很適合即時，因為它無法以良好的反應性修改被追蹤之組件的狀態。

此外，此類型的操作方式由於出現確認的工作，因此要求多工的環境。

此種配置周期性確認的操作方式可見於Java語言，確認作業配置一"垃圾收集器"類型的單元。

■ 關於硬體單元所從事之功能的細節

如前所述，組件的實施部分可對應於碼(它包括一或多個程式，以便完成組件的功能)，但是，也可以使用包括電子電路的硬體單元實施。

因此，組件C1所從事的功能可以使用電子電路實施，例如視訊解壓縮電路。不過，組件C1所從事的此項功能也可以只在軟體上實施。對一應用軟體而言，如果電子電路的確存在且能正確地操作，系統將使用組件C1。另一方面，如果電子電路不存在或它無法正確地操作，系統將使用組件C1'，使用方式對其它組件透明。在所有情況，都會從事功能。

同樣地，如果現在是以軟體碼從事功能，也可能稍後

五、發明說明（48）

是以使用電子電路的組件實施，其實施方式對其它組件完全透明。

圖9的圖說明組件C1被另一組件C'1取代，它們匯出的功能相同。

- 5 C1目前正在進行的執行如900所示，組件C2具有一與C1通信的通信通道，因此，將其鏈結到鏈結圖中請求GD在901以C'1取代C1。

接著，GD請求C1儲存它的目前狀態(步驟902)，C1在903從事此動作。接著，GD載入及初始化C'1(904及905)。

- 10 如果C'1的載入及初始化執行的不正確(步驟906)，GD返回一錯誤訊息給請求者C2。

如果C'1的載入及初始化發生的正確，GD在908將C1目前的狀態傳送給C'1，以使C'1將其本身置於相同的狀態。

- 15 基於這些目的，C'1解釋被傳送的目前狀態(909)，並接受它或拒絕它(910)。

如果是拒絕，GD卸載C'1(911)並將其去初始化(912)。接著，GD返回一錯誤訊息給請求者C2(913及914)。

- 20 如果現在C'1接受C1的目前狀態，GD更新它的鏈結圖(916)以C'1取代C1。接著，它為C'1重建通信通道(917)；這些通信通道原為C1使用。接著，GD將C1去初始化並卸載(918及919)。

替換已正確地執行，GD將其通知C2(920、921)；後者可以繼續它的處理。

以上所描述的是按照本發明之系統及其應用軟體的原

五、發明說明(49)

理，其它方面參考圖6。

容吾人特別指出，它也能參數化組件C1的介面，因此，當GD搜尋能匯出C1所要匯入之功能的組件時，它可在載入系統的一組組件中連續地搜尋，並將鏈結圖中每一個能匯出此功能之組件的"視野"傳送給C1，此步驟與圖6的步驟627等同。

按照參數化的選擇標準，C1能在被GD識別的這些組件間為其選擇適當的組件。

容吾人再特別指出，在它目前的型式中，按照本發明的系統最少需要4個組件才能操作，如圖10a所示(其中，箭頭指示組件間的功能傳送--因此，此類型的圖是鏈結圖的代表)。

這些組件如下：

- "互動"(interaction)，如前所述，需要它啟始系統的操作，
- "穿梭"(shuttle)，它被"互動"呼叫，且它本身呼叫以下兩個組件：
- GD，
- 最後是"VMM"，即所謂的記憶體組件或記憶體管理組件，它也呼叫GD。

圖10b同樣顯示一系統的最小組態，其中加入了3個組件"CPU"、"TIMER3"以及"SCHED"，以便允許多分緒管理。

最後，圖10c代表按照本發明之系統的增強型組態，它構成一小應用軟體。

五、發明說明(50)

在此圖中，"互動"組件使用一"COM"組件供通信，以便透過一網路交換資料，並使用"GUI"(圖形介面)組件顯示所要呈現的圖形。

"滑鼠"組件使其可以管理滑鼠周邊。它傳送關於改變此周邊之狀態的資訊(移動、點選按鈕)。
5 "GUI"組件使用"視窗"及"按鈕"組件執行"互動"組件所請求的顯示。

"視窗"及"按鈕"組件使用 GIF 及 JPEG 編碼解碼器組件在它們各自的顯示區域顯示影像。

GUI、COM、視窗、按鈕、GIF 及 JPEG 組件也使用系統之基本組件的功能。不過，在圖 10c 中並未顯示這些功能
10 對應的鏈結，以免此圖過於複雜。

各組件是分類到各域(domain)中，它們的分類並非與特權等級對應，而是按照使用的類型。以下是各種不同的域以及與其相關的組件：

15

域	描述
SCHEDULE	排程員組件 - MULTILEVEL
GESTOIN MEMOIRE	記憶體管理組件 - VMM - LMM
BUS	匯流排管理組件 - ISA - PCI
COMM	通信組件 - ETHERNET - FTP - IPSTACK

五、發明說明 (51)

	- SERIAL
INPUT	輸入系統管理組件 - EL TOUCH(觸控面板) - KEYMAP - PCKBD - PS2MOUSE(PS2滑鼠) - REBOOT(監視程式)
GFX	圖形介面管理組件 - GFX - BUTTON - WINDOW - SCROLLBAR - TEXT - IMAGE
SON	成音管理組件 - AUDIO編碼解碼器
IMAGES	影像及視訊管理組件 - JPEG編碼解碼器 - GIF編碼解碼器 - PNG編碼解碼器 - BMP編碼解碼器 - MPEG編碼解碼器
FILE SYSTEM (FS)	磁碟管理組件 - FAT - HDD - CD

須注意，由於按照本發明之系統的所有組件都是按照相同的格式實現，因此，系統是以相同的方法管理所有組件，且上述的優點一體適用於所有組件，無論它們的特權等級為何。

最後，很明顯，按照本發明的系統可以做到：

五、發明說明 (52)

- 程式設計師可以總覽作業系統的所有單元，無論它們的特權等級為何，
 - 使用相同的工具產生這些不同的單元，
 - 各不同部分間可以沒有限制地有效地相互連接。系統的
- 5 能力不在於它的組件數量，而是在於可能的相互連接，在傳統的系統中(單體及微核心)，各不同部分間的通信機構受到限制且固定。它必須有系統地通過無常的瓶頸(系統呼叫的數量受到核心的限制，無論是單體核心或是微核心等)。從屬的管理及通信機構的動態產生，使得可以克服此限制，
- 10
- 提供所有類型的碼相同的屬性。在本發明的例中，事實上不再需要設計一系統用以管理核心模組、程式庫或甚至應用軟體的"即插即用"。所有這些都由組件管理系統取代，
- 15
- 可以產生與未來所需完成之功能相同數量的組件。此允許按照本發明的系統可以完全地升級。

此外，如前所述，經由在很多情況返回錯誤訊息，本發明給予發展者極大的可讀取性及反應性。

- 事實上，在組件的載入期間，自動地為此組件執行一
- 20 函式(Init)。此函式(Init)確認載入之組件操作時所需的組件組確實已載入系統。如果此項確認不符合要求(至少有一個請求的功能沒有完成，或是，與能提供所請求之功能之組件相關的硬體單元有瑕疵等)，即返回一錯誤訊息給呼叫方。

五、發明說明 (53)

此外，如果被呼叫的組件與一個有瑕疵的單元(硬體及/或軟體)有關，它自動卸載到系統的隨機存取記憶體中。

此外，從圖9中可看出，當組件被呼叫時，錯誤訊息也會返回。

- 5 因此，如果與操作發出呼叫之組件相關的硬體或軟體單元不存在或有瑕疵，錯誤訊息會保持它的建議，不需要系統做完整的測試操作。

(請先閱讀背面之注意事項再填寫本頁)

裝

訂

線

四、中文發明摘要(發明之名稱： 資訊系統、模組及其方法)

本發明與用以發展意欲由電腦系統實施之電腦計畫的方法有關，其方法包括功能分析階段，其目的是將所要提供的功能分類，在電腦機構的協助下產生計畫之組件的階段，以便完成該功能，宣告階段，在同一系統上宣告產生階段期間所產生的組件，以及載入階段，將該組件載入系統的記憶體區供系統實施它們，其特徵為：

- 在組件產生階段期間，每一個功能完全由與其相關的組件完成，
- 10 • 在組件宣告階段期間，將每一個被產生的組件宣告給系統以載入組件，以及
- 在組件載入階段期間，提供機構(GD)結合被載入之組件所請求的每一項功能以自動搜尋能提供該功能的組件。

訂

63

四、中文發明摘要(發明之名稱: Systeme informaque madulaire et procede)
associe

The invention relates to a process for developing computer projects intended to be implemented by a computer system, process comprising a phase of functional analysis with a view to cataloguing the functionalities which are to be catered for, a phase of creating components of the project with the aid of computer means so as to fulfil the said functionalities, a phase of declaration on the same system of the components created during the creation phase, and a phase of loading the said components into a memory area of the system for their implementation by the system,

characterized in that:

- during the components creation phase, each functionality is completely fulfilled by the component associated therewith,
- during the components declaration phase, each component created is declared to a system loading component, and
- during the components loading phase, there are provided means for, in association with each functionality requested by a loaded component, automatically searching for a component offering the said functionality.

(請先閱讀背面之注意事項再填寫本頁各欄)

裝

訂

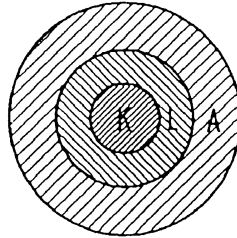


圖 1

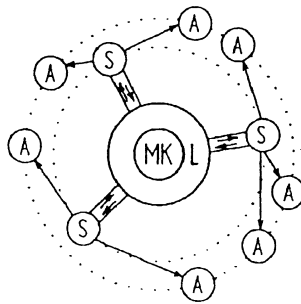


圖 2

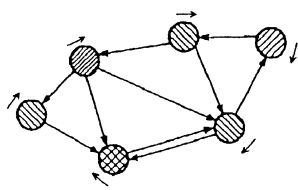


圖 3a

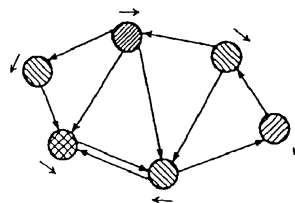


圖 3b

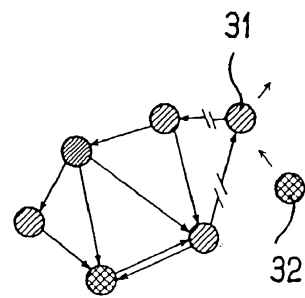


圖 3c

2 / 7

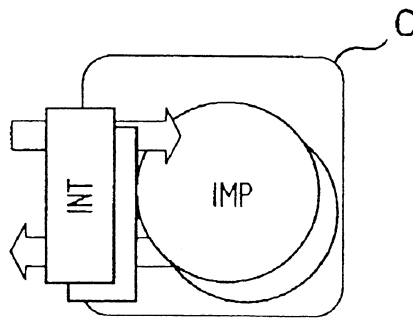


圖 4

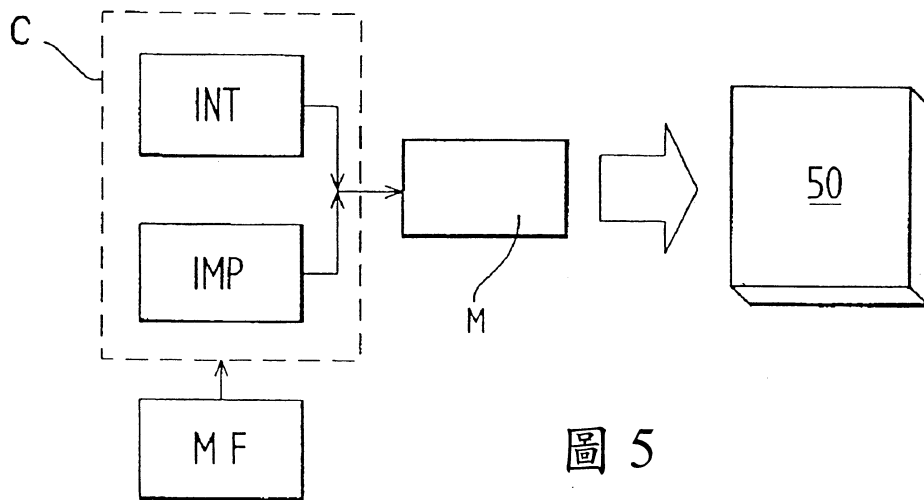


圖 5

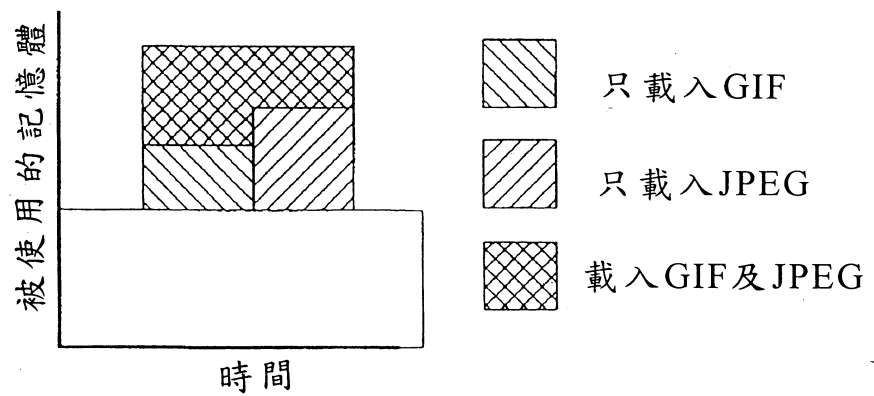


圖 8

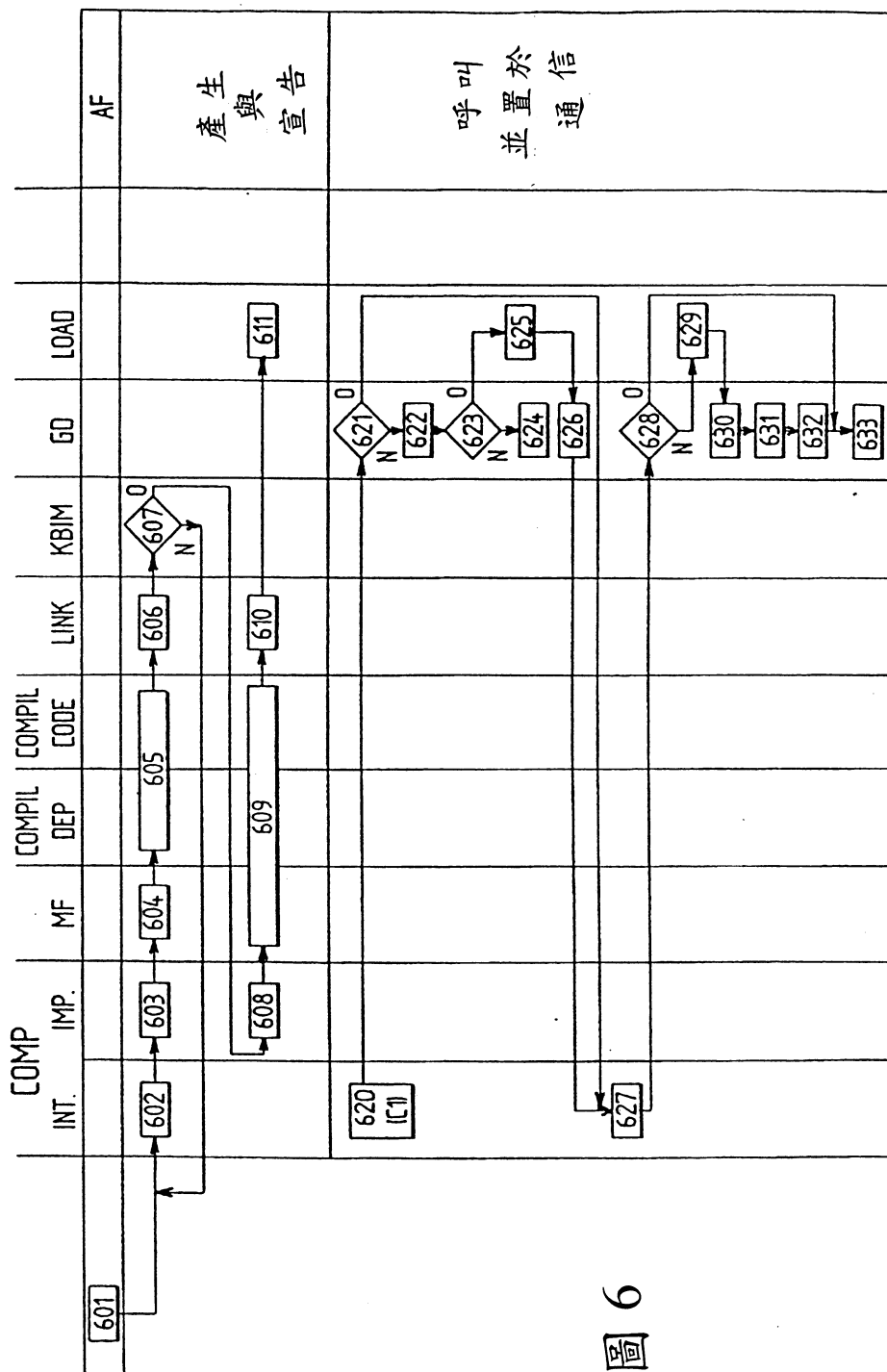


圖 6

4 / 7

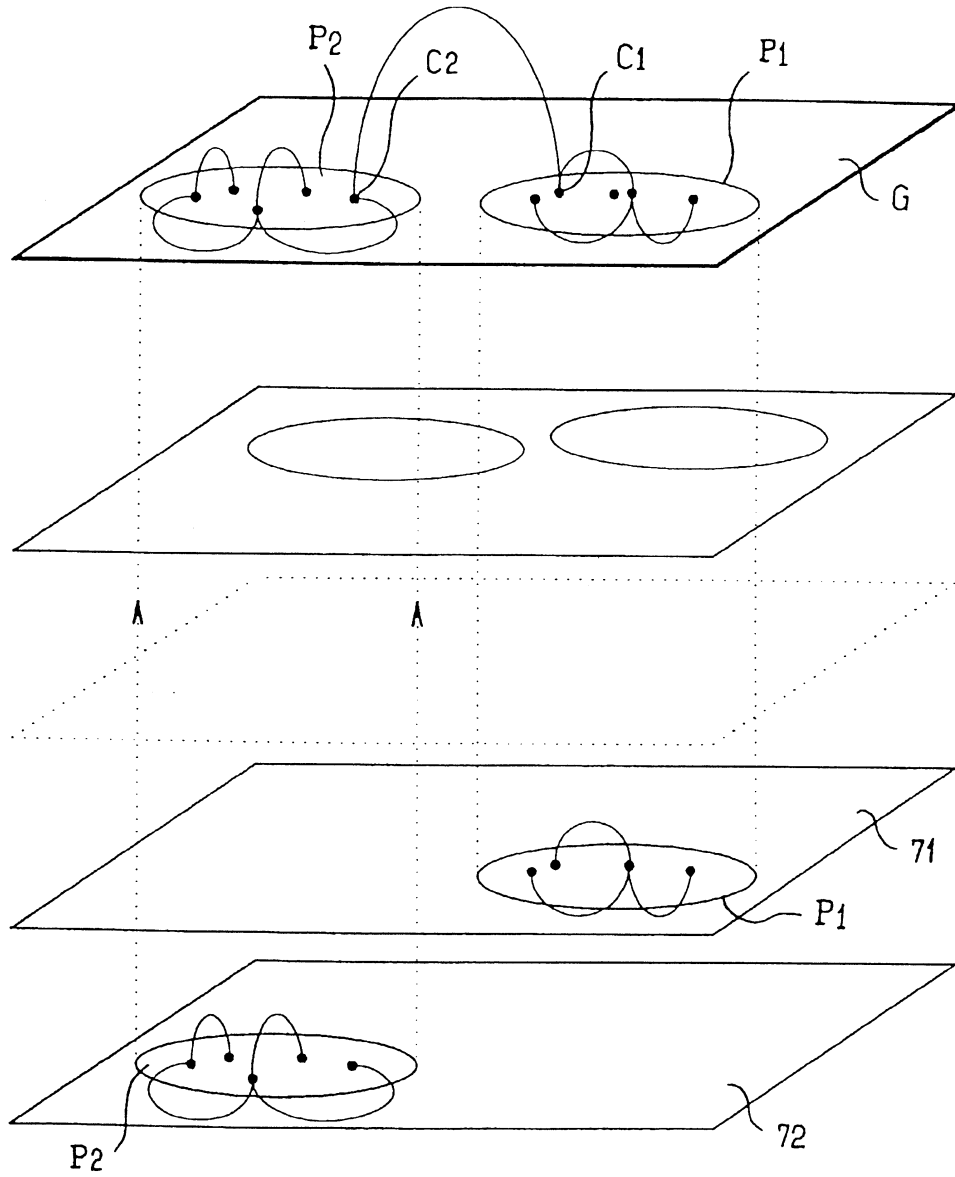


圖 7

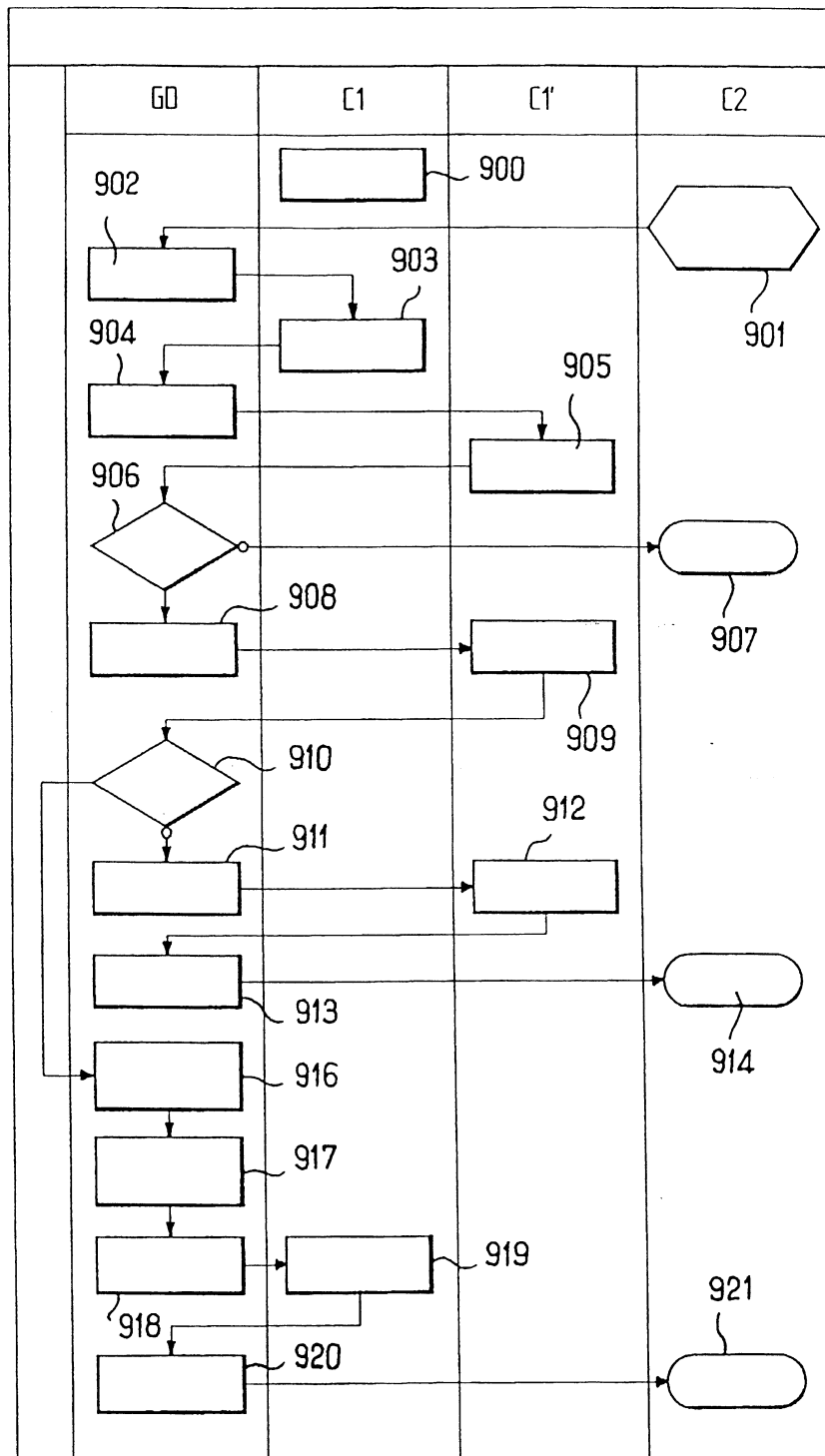


圖 9

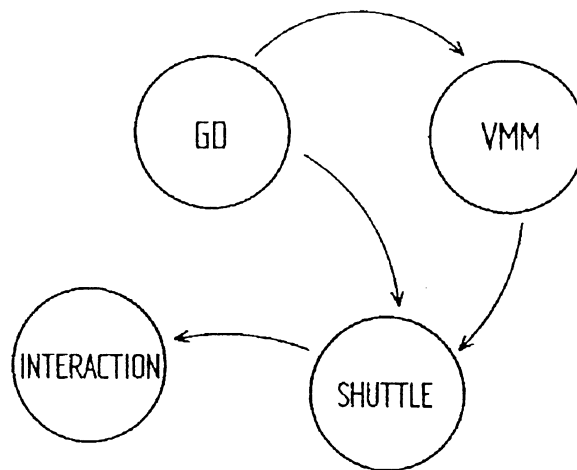


圖 10a

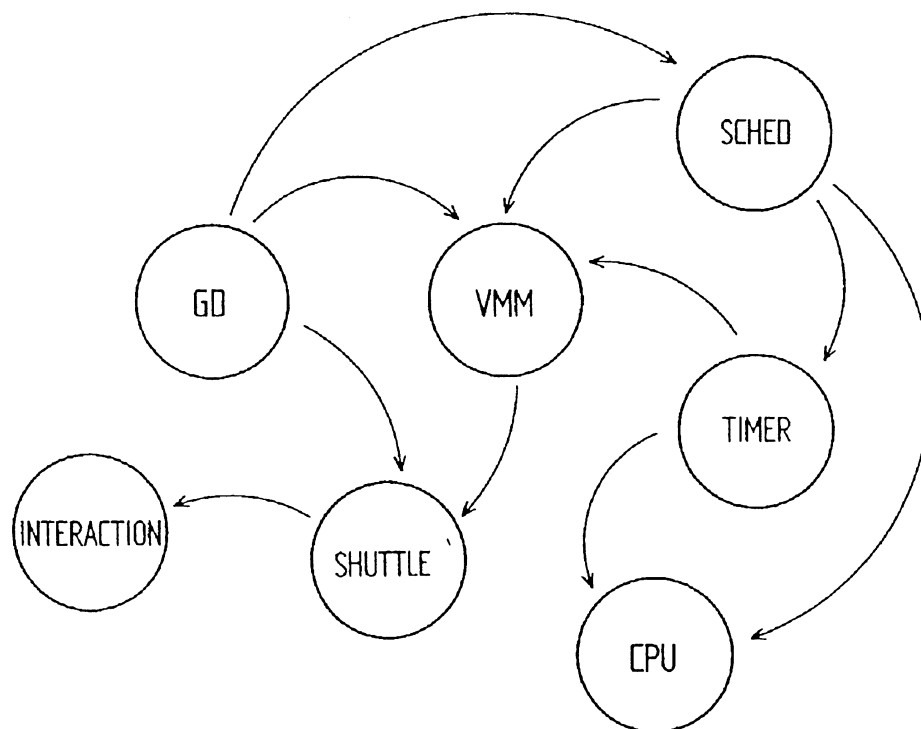


圖 10b

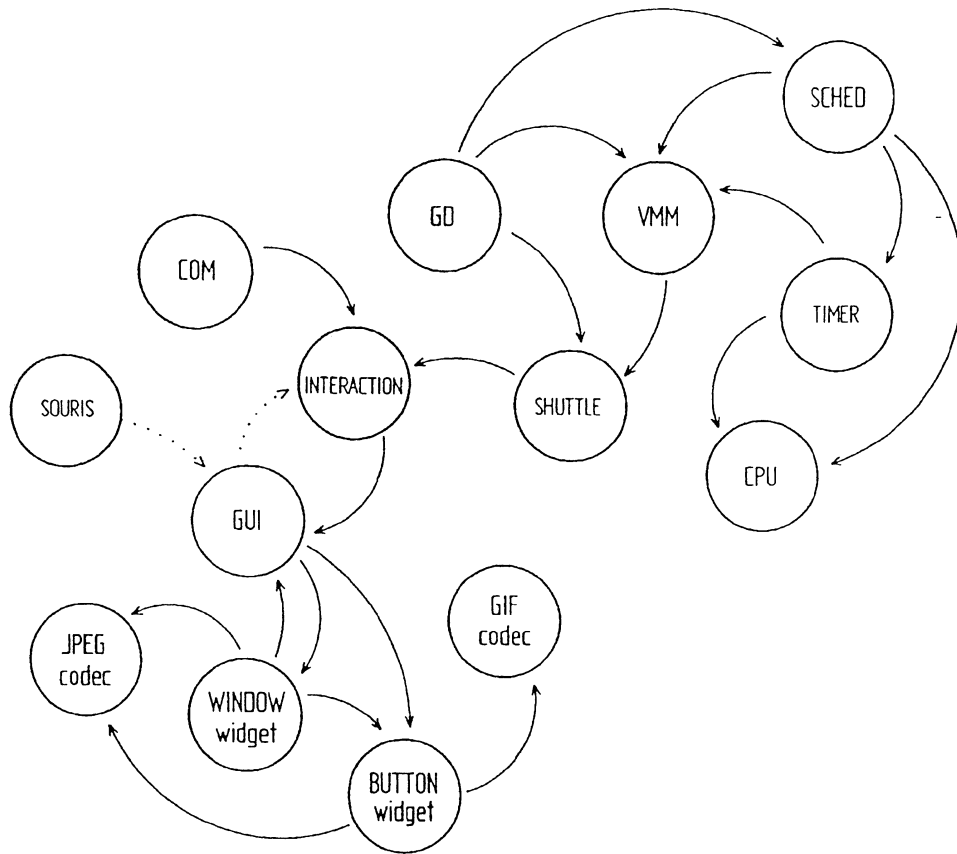


圖 10c

六、申請專利範圍

專利申請案第 90114791 號

ROC Patent Appln. No. 90114791

修正之申請專利範圍中文本 - 附件(一)

Amended Claims in Chinese - Encl. (I)

(民國 91 年 10 月 25 日送呈)

(Submitted on October 25, 2002)

1. 一種用以發展意欲由電腦系統實施之電腦計畫(P1, P2)的方法，其方法包括功能分析階段(601)，其目的是將所要提供的功能歸類，在電腦機構的協助下產生計畫之組件(C1, C2)以便完成該些功能的階段，在產生階段期間產生組件的同一系統上宣告的階段，以及，將該組件載入系統的記憶體區供系統實施它們的載入階段，其特徵為：
 - 在組件產生階段期間，每一個功能完全由與其相關的組件完成，
 - 在組件宣告階段期間，將每一個被產生的組件宣告給系統以載入組件，以及
 - 在組件載入階段期間，提供機構(GD)結合被載入之組件所請求的每一項功能以自動搜尋能提供該功能的組件。
2. 如前述申請專利範圍第1項中的方法，其特徵為，執行該自動搜尋時將條件列入考慮。
3. 如前述申請專利範圍第2項中的方法，其特徵為，該條件是以與每一個組件個別相關之屬性的函式所表示之標準的函式表示。
4. 如前述申請專利範圍第1至3項中任一項的方法，其特

修正本局
91年10月25日
所提之
修正
委員明示
准予修正。

裝

計

線

六、申請專利範圍

01/10/25

徵為，在組件的產生期間，執行以下的步驟：

- 產生(602)組件的介面原始檔，其目的是為構成組件的"介面"部分(INT)；
 - 產生(603)至少一個用以實現組件功能的碼原始檔，其目的是為構成組件的"實施"部分(IMP)。
5. 如前述申請專利範圍第4項中的方法，其特徵為，所產生的每一個碼原始檔其目的是為構成組件的"實施"部分，可以由任何程式語言有系統地陳述。
6. 如前述第1至3項申請專利範圍其中任一項的方法，其特徵為，經由配置軟體及/或硬體機構，每一個組件可以完成與其相關的功能。
7. 如前述第4項申請專利範圍中的方法，其特徵為，每一個組件的介面檔包括對該組件所要完成之功能的描述，與組件完成此功能的方法無關。
8. 如前述申請專利範圍第7項中的方法，其特徵為，介面檔也包括該組件為完成其本身功能需要從其它組件匯入之功能的描述。
9. 如前述申請專利範圍第1至3項中任一項的方法，其特徵為，每一個組件(C1)可以被另一個能匯出相同功能的組件(C'1)動態地取代。
10. 如前述申請專利範圍第9項中的方法，其特徵為，該取代的配置按如下的步驟：
- 儲存要被取代之組件的目前狀態，
 - 載入及初始化新組件，

六、申請專利範圍

91年0月25日

修正

補充

- 建立呼叫功能之組件與新組件間的通信通道，

- 卸載被取代的組件。

11. 如前述申請專利範圍第10項中的方法，其特徵為，如果新組件的載入或初始化沒有正確地執行，提供的機構將返回一錯誤訊息。

12. 如前述第8項申請專利範圍中的方法，其特徵為，在組件的產生期間，也要執行以下的步驟：

- 產生(604)製作檔，其中明確說明：
 - ✓ 在產生碼原始檔的步驟期間用來產生原始檔的語言，
 - ✓ 碼原始檔對應於組件的"實施"部分
- 產生(605)組件的原型，
- 編輯組件之"介面"部分與"實施"部分間的鏈結(606)，以便產生可執行的組件原型，
- 配置製作檔(609)，以便個別地編譯組件，並為組件之"介面"部分的介面原始檔以及為組件之"實施"部分的每一個碼原始檔構成各自的物件檔，
- 將前步驟期間所產生的物件檔包封(610)成單一的可執行模組。

13. 如前述申請專利範圍第12項中的方法，其特徵為，在產生組件的原型期間，其原型包括產生：

- 物件介面，由編譯組件之介面原始檔所產生，
- 原型物件檔，與組件之"實施"部分的每一個碼原始檔有關。

六、申請專利範圍

14. 如前述申請專利範圍第13項中的方法，其特徵為，在編譯組件之介面原始檔期間，要執行：
- 分析與確認介面之原始檔的一致性，
 - 將介面的原始檔轉換成二進位資料，
 - 自動產生至少一個檔，以便呼叫要被組件匯入的功能。
15. 如前述申請專利範圍第12項中的方法，其特徵為，在編輯組件之"介面"部分與"實施"部分間的鏈結，以便產生可執行的組件原型步驟之後，執行自動的完整性測試步驟(607)，在此期間，確認計畫是否前後一致與封閉。
16. 如前述申請專利範圍第15項中的方法，其特徵為，在自動的完整性測試期間，在可存取之組件的表中，為計畫中所要求的每一個功能自動搜尋能完成該功能的組件。
17. 如前述申請專利範圍中第16項的方法，其特徵為，如果識別出有數個組件能完成所請求的功能，可以在這些組件間選擇一特定的組件。
18. 如前述第16項申請專利範圍中的方法，其特徵為，如果完整性測試顯示計畫未正確地封閉，發出一錯誤訊息用以指示不一致性，以便能修改某些介面及/或增加某些介面。
19. 如前述第8項申請專利範圍中的方法，其特徵為，對每一個被載入系統的組件而言，對於需要從其它組件匯

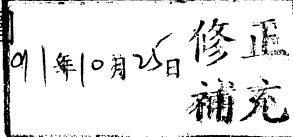
六、申請專利範圍

9/10月25日修正
補充

入以便完成其本身功能的描述，永遠可被自動組件搜尋的該機構(GD)存取。

20. 如前述第8項申請專利範圍中的方法，其特徵為，在將前一步驟所產生的物件檔包封(610)到單一個可執行模組中的期間，要確認計畫之各不同介面部分供應的所有功能都已在相關的原始檔中確實地被完整地描述且已編譯。
21. 如前述第8項申請專利範圍中的方法，其特徵為，其方法包括將儲存在系統之隨機存取記憶體區中之組件的映圖(G)載入系統。
22. 如前述第21項申請專利範圍中的方法，其特徵為，該映圖意欲供發出呼叫之組件在呼叫功能時查閱，以便決定該映圖的組件是否能供應所呼叫的功能。
23. 如前述第22項申請專利範圍中的方法，其特徵為，自動組件搜尋的該機構(GD)可查閱該映圖。
24. 如前述第21項申請專利範圍中的方法，其特徵為，在系統操作期間該映圖即時更新，是載入系統之組件的函式。
25. 如前述申請專利範圍第1至3項中任一項的方法，其特徵為，每一個組件可與一特權等級相關，為某一組件存取其它組件之可能性的條件，是該其它組件各自之特權等級的函式。
26. 如前述申請專利範圍第1至3項中任一項的方法，其特徵為，提供卸載每一個組件的機構，當電腦計畫的操

六、申請專利範圍



作不再需要其所提供的功能時將其卸載。

27. 如前述第26項申請專利範圍中的方法，其特徵為，該機構包括每一個組件的使用計數器。
28. 如前述第27項申請專利範圍中的方法，其特徵為，該計數器計算匯入結合有計數器之組件之功能之組件的數量，且一旦該組件的使用計數器遞減到零時它即被卸載。
29. 如前述第28項申請專利範圍中的方法，其特徵為，提供一機構(GD)，用以自動釋放該被卸載之組件使用的資源。