(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2007 年6 月14 日 (14.06.2007)

PCT

(10) 国际公布号
WO 2007/065357 A1

(51) 国际专利分类号:
G06F 17/30 (2006.01)

(21) 国际申请号: PCT/CN2006/003297

(22) 国际申请日: 2006 年12 月5 日 (05.12.2006)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
200510126683.6
2005 年12 月5 日 (05.12.2005) CN
200510131072.0
2005 年12 月9 日 (09.12.2005) CN

(71) 申请人 (对除美国外的所有指定国): 北京书生国际信息技术有限公司(BEIJING SURSEN CO.,LTD) [CN/CN]; 中国北京市海淀区学院路35号世宁大厦13层, Beijing 100083 (CN)。

(72) 发明人; 及

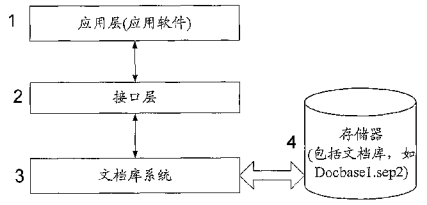
(75) 发明人/申请人 (仅对美国): 王东临(WANG, Donglin) [CN/CN]; 中国北京市海淀区学院路35号世宁大厦13层, Beijing 100083 (CN)。 郭旭(GUO, Xu) [CN/CN]; 中国北京市海淀区学院路35号世宁大厦13层, Beijing 100083 (CN)。 刘昌伟(LIU, Changwei) [CN/CN]; 中国北京市海淀区学院路35号世宁大厦13层, Beijing 100083 (CN)。 姜海峰(JIANG, Haifeng) [CN/CN]; 中国北京市海淀区学院路35号世宁大厦13层, Beijing 100083 (CN)。

(74) 代理人: 北京德琦知识产权代理有限公司(DEQI INTELLECTUAL PROPERTY LAW CORPORATION); 中国北京市海淀区知春路1号学院国际大厦7层, Beijing 100083 (CN)。

[见续页]

(54) Title: DOCUMENT PROCESSING SYSTEM AND METHOD

(54) 发明名称: 文档处理系统和方法



1 APPLICATION LAYER(APPLICATION SOFTWARE)
2 INTERFACE LAYER
3 DOCUMENT BASE SYSTEM
4 STORAGE DEVICE

(57) Abstract: The present invention discloses a document processing system and method. The system includes an application layer and a document base system, the application layer comprises at least one application software for issuing the standard instructions which contain actions and objects and process the document to the document base system; the document base system for implementing the actions indicated by the standard instructions to the objects in the stored document data. The system and method of the present invention separate the application layer from the data processing layer to implement the mutual operations of the documents.

(57) 摘要:

本发明公开了一种文档处理系统和方法，该系统包括应用层和文档库系统，所述应用层包括至少一个应用软件；应用软件，用于向文档库系统发出包含动作和对象的、对文档进行操作的标准指令；文档库系统，用于对所存储的文档数据中的所述对象，执行标准指令指示的动作。本发明的这种系统和方法将应用层和数据处理层分离，实现文档互操作。

WO 2007/065357 A1

(81) 指定国（除另有指明，要求每一种可提供的国家保护）：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, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW。

(84) 指定国（除另有指明，要求每一种可提供的地区保护）：ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), 欧洲 (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)。

本国际公布：

— 包括国际检索报告。

所引用双字母代码及其它缩写符号，请参考刊登在每期PCT公报期刊起始的“代码及缩写符号简要说明”。

文档处理系统和方法

技术领域

本发明涉及一种文档处理系统和方法。

发明背景

信息可大致分为结构化数据和非结构化数据，其中以书面文档和流媒体为主的非结构化数据根据资料统计占有量超过百分之七十。结构化数据的结构比较简单，即一个二维表结构，其处理技术以数据为代表，主要是利用数据库系统进行处理，从上世纪七八十年代开始发展，到九十年代达到顶峰，研发和应用已经比较成熟。非结构化数据则没有固定数据结构，因此对非结构化数据的处理非常的复杂。

目前处理各种非结构化文档的软件已经比较普及，形成了多种文档格式林立的状况。例如，文档编辑目前就存在 Microsoft 的 word、WPS、永中的 Office、Red 的 Office 等。通常，一个内容管理软件往往要处理二三百种文档格式，而且这些格式还在不断更新，给这类软件的开发带来了巨大的困难。如何解决文档通用性、进行数字内容提取、格式兼容越来越成为人们的关注点，人们迫切希望解决以下问题:

1）文档不通用:

基本上，不同用户只能交换同一种软件处理的文档，无法交换不同软件处理的文档，形成信息封闭。

2）访问接口不统一、数据兼容代价太高:

不同的文档处理软件之间，文件格式互不兼容，在处理过程中要么利用对方组件解析（前提是对方提供相应接口），要么自己投入研发力量从头到尾的解析对方的格式。

3）信息安全较差:

目前针对书面文档的权限控制手段单一，主要是数据加密、口令认证。因为信息泄露，每年造成巨大损失的公司案例层出不穷。

4）都是针对单个文档的处理，缺乏多文档管理手段:

每个人电脑中都有大量文档，但多个文档之间缺乏有效的组织管理，而且资源共享很难。如，字库/字体文件、全文数据检索等。

5）页面分层的技术不完善:

目前一些软件，如 Adobe 的 photoshop，Microsoft 的 word，多多少少已经有层的概念，但层的功能还比较单一，管理手段比较简单，不能满足应用需求。

6）检索手段不够丰富:

随着信息的海量化，用任何一个关键词来搜索都会得到数量庞大的检索结果，全文检索技术基本解决了查全率的问题，但查准率迅速上升为首要问题。现有技术还没有很充分地利用全部信息来解决查准率问题，例如每个文字的字体、字号完全可以用来判断该文字的重要性，但都在检索时被忽略了。

虽然各大公司目前都努力将自己特有的文档格式发展为市场标准，各标准组织也致力于制订通用的文档格式标准。但不管是专有的文档格式（如.doc）还是开放的文档格式（如 PDF），只要是以文档格式为标准，就不可避免产生以下问题:

a）重复开发，效果不统一:

使用同一标准的不同软件都需要自己去解释、生成该格式的文档，造成大量重复开发，而且会因为各家解释程序不同，例如有的完善有的相对简单，有的支持新版本有的只支持旧版本数据，同一文档在不同软件下显现出不同的版式，甚至出现解释错误导致无法打开文档。

b）阻碍创新:

软件是不断创新的行业，但由于每增加一个新功能就需要增加描述该功能的信息，而且只有等到标准修订的时候才能增加新的格式，因此把存储格式固定死，将会妨碍技术创新的竞争。

c）影响检索性能:

对海量信息，需要增加大量的检索信息以提高检索性能，但固定死的存储格式难以增加检索信息

d）影响可移植性和可伸缩性:

在不同的系统环境下，不同的应用需求，可能会有不同的存储要求。例如，存储在硬盘上就需要考虑如何减少磁头寻道的次数以提高性能，而在嵌入式应用中数据都相当于存储在内存中的，就不存在这个问题。例如，同一个厂商的数据库软件在不同平台上就可能会使用不同的存储格式。因此，设置文档存储标准将会影响系统的可移植性和可伸缩性。

现有技术中最开放、可交换性最好的文档是 Adobe Acrobat 的 PDF。然而，虽然 PDF 已经成为全球文档分发、交换的事实标准，但也不能实现在不同的软件之间交换 PDF 文档，也就是说，不能实现 PDF 文档的互操作性。而且，无论是 Acrobat 还是 Office，都只能对单文档进行处理，缺乏对多文档的管理功能，不具备对文档库进行操作的功能。

另外，在文档信息安全的方面，现有技术也存在较多缺陷。Word 和 PDF 这些应用最广泛的文档，都是采用对数据加密或者口令认证等进行数据安全控制，没有提供系统的身份认证机制，对权限的控制都是整个文档范围的，不能细化到文档内的任意区域，无法对任意逻辑数据设定加密和签名。现有的内容管理系统虽然能够提供较好的身份认证机制，但由于与文档处理系统是分离的，不仅管理粒度只能做到文档级，而且无法在文档使用过程中对文档实施安全控制，难以进行必要的安全

管理。由此可见，由于现有的安全机制与文档处理是分离的模块，容易出现安全缝隙。

发明内容

本发明提供了一种文档处理的系统和方法，实现对文档的互操作、对多文档的管理、更好的文档安全性以及更好的查询检索。

一种文档处理系统，该系统包括应用层和文档库系统，所述应用层包括至少一个应用软件；

应用软件，用于向文档库系统发出包含动作和对象的、对文档进行操作的标准指令；

文档库系统，用于对所存储的文档数据中的所述对象，执行标准指令指示的动作。

一种文档处理方法，该方法包括：

应用软件向文档库系统发出包含动作和对象的、对文档进行操作的标准指令；

文档库系统对所存储的文档数据中的所述对象，执行标准指令指示的动作。

一种文档处理系统，该系统包括应用层、文档库系统和接口层，所述应用层包括至少一个应用软件；

应用软件，用于通过接口层，向文档库系统发出包含所述动作和对象的、对文档进行操作的标准指令；

文档库系统，用于通过接口层接收该标准指令，对所存储的文档数据中的对象执行标准指令指示的动作。

本发明改变了从用户界面到文档存储都由一个软件来完成的现状，将其划分为应用层和文档库系统层，并定义一个规范两层之间交互的接

口标准，还可以进一步构建一个符合该接口标准的接口层。文档库系统是具备各种文档操作功能的通用技术平台，应用软件要对文档进行操作时就通过该接口层来向文档库系统发出相应指令，文档库系统根据该指令执行相应操作。这样，只要各应用软件和各文档库系统都遵循同样的标准，不同应用软件就可以通过同一个文档库系统对同一文档操作，即可实现对文档的互操作。同样，同一个应用软件也可以通过不同文档库系统对不同文档进行操作，而不用分别对每种文档格式都进行单独开发。

本发明还包括一个通用文档模型，该模型能与各应用软件所需要处理的文档相符合。接口标准就是基于该文档模型来确定的，这样才能实现不同的应用软件都可以通过接口层来对文档进行操作。该通用文档模型也适用于各种文档格式，这样同一个应用软件才可以通过接口层来对不同文档格式进行操作。

接口标准定义了基于该通用文档模型对文档进行操作的各种指令，以及应用软件向文档库系统发送指令的方式。文档库系统具备实现这些指令的功能，以供应用软件调用。

该通用文档模型还包括由多个文档组成的文档集、文档库和文档仓库等层次，接口标准中也包含对多文档的组织管理、查询检索、安全控制等指令。

该通用模型还包括将页由具有上下顺序的层组成，接口标准中也包含对层的各种操作指令，以及对一个文档某一层所对应源文件的存储和提取。

文档库系统还具备对文档的信息安全管理控制功能，如基于角色的细粒度权限管理，并在接口标准中定义了相关的操作指令。

依照本发明，使得应用层和数据处理层分离。这样应用软件不再直

接跟具体的文档格式打交道，文档也不再与特定应用软件绑定，从而使得同一文档能在不同的应用软件之间通用，同一应用软件也能对不同文档进行操作，实现了文档的互操作；整个文档处理系统还具备多文档处理功能，而不局限在单文档处理；将页分成多层后，可以实现对不同层实施不同管理和控制，更便于不同应用软件对同一页的操作（可以设计成不同应用软件管理和维护不同层），为以源文件方式进行编辑提供了便利，也是一种很好的保留历史痕迹的方式；通过将信息安全集成在文档处理的核心层，可以消灭安全缝隙，还能使安全机制与文档操作紧密地结合为一体，而不是可以分离的两个模块，同时有更多的空间部署安全管理技术，相关代码也能隐藏得更深，能更有效地防御非法攻击，提高安全可靠度，另外还能提供细粒度的安全管理手段，如更多的权限类别，更小的管理单元。

附图简要说明

图 1 为依照本发明的文档处理系统的结构框图。

图 2 示出了依照本发明一优选实施例的通用文档模型的组织结构。

图 3 示出了图 2 所示通用文档模型中文档库对象的组织结构。

图 4 示出了图 3 所示文档库对象中文档库辅助对象的组织结构。

图 5 示出了图 3 所示文档库对象中文档集对象的组织结构。

图 6 示出了图 5 所示文档集对象中文档对象的组织结构。

图 7 示出了图 6 所示文档对象中页面对象的组织结构。

图 8 示出了图 7 所示页面对象中层对象的组织结构。

图 9 示出了图 8 所示层对象中版面对象的组织结构。

图 10-17 为本发明的实施例中所定义的动作。

图 18 为以 UOML 接口为例的文档处理系统的处理示意图。

实施本发明的方式

以下结合附图及实施例，对本发明进行进一步详细说明。应当理解，此处所描述的具体实施例仅仅用于解释本发明，并不用于限定本发明。

如图 1 所示，依照本发明的文档处理系统主要包括应用软件、接口层、文档库系统和存储设备。

其中的应用软件包括现有的任何文档处理和内容管理软件，这些应用软件都位于文档处理系统的应用层，通过发送符合接口标准的指令来对文档进行操作，所述操作都是对符合通用文档模型的文档进行的，与具体存储格式无关。

其中的接口层符合规范应用层和文档库系统之间交互的接口标准，所述应用层通过接口层向文档库系统发送标准指令，所述文档库系统通过接口层向应用层返回执行的结果。由此可见，由于应用软件均可以通过接口层发出标准指令，对符合通用文档模型的文档进行操作，所以不同的应用软件可以通过同一文档库系统对同一文档进行操作，同一应用软件也可以通过不同文档库系统对不同格式的文档进行操作。

优选地，接口层可包括上接口单元和下接口单元，应用层通过上接口单元发送标准指令至下接口单元，文档库系统通过下接口单元接收标准指令，下接口单元还用于将文档库系统的执行结果通过上接口单元返回给应用系统。在实现上，上接口单元可位于应用层中，下接口单元可位于文档库系统中。

其中的文档库系统为文档处理系统的核心层，根据应用软件通过接口层发来的标准指令执行具体的文档处理操作。

其中的存储设备为文档处理系统的存储层，常用的是硬盘或者内存，也可以是光盘、闪存、软盘、磁带，也可以是远程的存储设备，总之只要具备数据的存储能力即可。在存储设备中存储有多个文档，但对

应用软件而言并不需要关心文档的具体存储方式。

由此可见，依照本发明，使得应用层和数据处理层真正分离开来，文档不再与特定应用软件绑定，应用软件不再直接跟具体的文档格式打交道，不同的应用软件可以对符合通用文档模型的同一文档进行编辑，使不同应用软件之间具有良好的文档互操作性。

本发明还公开了一种应用软件，其包括用于发出标准指令的接口单元，所述标准指令用于对符合通用文档模型的文档进行操作。

本发明还公开了一种文档库系统，其包括：用于接收标准指令的接口单元；和，用于根据该标准指令对符合通用文档模型的文档进行操作的处理单元。

本发明还公开了一种接口层，其包括：

上接口单元，用于发送对符合通用文档模型的文档进行操作的标准指令；

下接口单元，用于接收该标准指令。

进一步，上接口单元可以根据应用层发出的指令生成标准指令，下接口单元判断接收到的指令是否符合标准，并解析符合标准的指令。

以下对本发明的文档处理系统的具体实施方式进行阐述。

通用文档模型

可参考纸张的特性定义所述通用文档模型，这是因为以纸张作为文档信息的记录手段是通行至今的标准方法，只要能具备纸张的所有功能，就能满足工作、生活等实际应用的需求。

如果把文档中的一页当成一张纸，凡是能画到纸上的就记录下来，该通用文档模型能够描述页面上的所有可见内容。现有技术中的页面描述语言（如 PostScript）可以描述所有能印在纸上的信息，因此这一部分就不再详细阐述。一般说来，页面上的可见内容最终都可以归为文字、

图形、图像三类。

如果文档中涉及到特定字体或特殊字符的话，为了保证在各台电脑上都能有相同的效果，就需要在文档中嵌入相应字库。为了提高存储效率，字库资源应当共享，这样即使在多处使用了同一字符，也只需要嵌入一个字库。图像有时也是可能在多处出现的，例如每一页共同的底图，或经常出现的公司标识，这种情况下最好也能共享这些图像。

当然，作为更加先进的信息处理工具，不能仅仅模拟纸张的特性，还需要增加一些增强的数字特性，例如元数据、导航、导读、微缩版面。元数据是描述数据的数据，例如作者、出版社、出版时间、ISBN 号等就是图书的元数据。元数据是业内通用名词，也不在此赘述。导航是类似图书目录的信息，也是业内通用名词。导读信息描述了一篇文章所在的区域和阅读顺序，这样当阅读者读完一屏后就可以根据该信息自动判断下一屏应该显示什么，这样还能做到自动换栏、自动转版，而不用阅读者再手工指定位置。微缩版面是事先生成的各页面的微缩图，阅读者可以通过查看微缩版面来指定阅读哪一页。

图 2 是本发明的一优选实施例的通用文档模型。如图 2 所示，该通用文档模型包含文档仓库、文档库、文档集、文档、页、层、对象组、版面对象等多个层次。

其中，文档仓库由一个或多个文档库组成，文档库之间的关系相对于文档库之下的层次之间的关系相对要松散一些，文档库之间可以非常简单地组合和拆离，而不用对文档库本身的数据做改动，该多个文档库之间往往没有建立统一索引（特别是全文索引），很多对文档仓库的检索操作一般都需要遍历各文档库的索引，而没有统一的索引可用。每个文档库由一个或多个文档集组成，每个文档集由一个或多个文档组成，还可以包含任意数量的子文档集。这里所说的文档相当于目前普通的一

个文档文件（例如 DOC 文档），通用文档模型可以规定一个文档只能属于一个文档集，但也可以允许一个文档属于多个文档集。文档库不是多个文档的简单组合，它把多个文档紧密地组织起来，特别是为文档内容统一建立了各种检索索引后就能带来更大的便利性。

每个文档由一页或存在一定顺序（如前后顺序）的多页组成，每页的版心可以不同，而且版心也不一定是矩形的，可以是任意形状，可以用一条或多条封闭曲线表示版心。

每页又由一层或按一定顺序（如上下顺序）的多层组成，各层之间如同玻璃板的叠加关系。层由任意数量的版面对象和对象组组成，版面对象是指状态（如字体、字号、颜色、ROP 等）、文字（包括符号）、图形（如直线、曲线、填充了指定颜色的闭合区域、渐变色等）、图象（如 TIF、JPEG、BMP、JBIG 等）、语义信息（如标题开始、标题结束、换行等）、源文件、脚本、插件、嵌入式对象、书签、链接、流媒体、二进制数据流等。一个或多个版面对象可以组成一个对象组。对象组也可以包含任意数量的子对象组。

文档库、文档集、文档、页、层都可以还包括元数据（如名称、最后修改时间等，其类型可以根据应用需求来设置）和／或历史痕迹；文档中还可以包括导航信息、导读信息、微缩版面；也可以把微缩版面放在页或者层这个层次；文档库、文档集、文档、页、层、对象组都可以还包括数字签名；语义信息最好跟着版面信息走，这样可以避免数据冗余，也比较容易与版面建立对应关系；文档库、文档还可以包括字库、图像等共享资源。

该通用文档模型还可以定义一个或多个角色，为每个角色分配一定权限。权限以文档库、文档集、文档、页、层、对象组、元数据为单元进行分配，定义每个角色对该单元是否可读、是否可写、是否可复制、

是否可打印，等等。

该通用文档模型是一个超越以往单个文档对应单个文件的方式，文档库中包含多个文档集、文档集中包含多个文档，而对于文档库中文档内容，采用了细粒度的访问和安全控制，可以具体访问文档库中某个文字或者矩形，而不像现在的文档管理系统只能访问到文件名。

图 3 至图 9 示出了本发明一优选实施例的通用文档模型所涉及的各对象的组织结构示意图。所述的各对象的组织结构是树状结构，是逐层展开、细化的。

文档仓库对象是由一个或多个文档库对象组成（图中未示出）。

如图 3 所示，文档库对象包括一个或多个文档集对象、任意数量文档库辅助对象和任意数量的文档库共享对象。

如图 4 所示，所述的文档库辅助对象包括元数据对象、角色对象、权限对象、插件对象、索引信息对象、脚本对象、数字签名对象、历史痕迹对象等。文档库共享对象是指文档库中的不同文档可能共享的对象，如字库对象、图像对象等。

如图 5 所示，每个文档集对象包括一个或多个文档对象、任意数量的文档集对象和任意数量的文档集辅助对象。文档集辅助对象包括元数据对象、数字签名对象、历史痕迹对象。当文档集对象包括多个文档集对象时，其类似于资源管理器中的文件夹包括多个文件夹的形式。

如图 6 所示，每个文档对象包括一个或多个页面对象、任意数量的文档辅助对象和任意数量的文档共享对象。文档辅助对象包括元数据对象、字库对象、导航信息对象、导读信息对象、微缩版面对象、数字签名对象、历史痕迹对象等。文档共享对象包括文档中的不同页面可能共同使用的对象，如图像对象、印章对象等。

如图 7 所示，每个页面对象包含一个或多个层对象和任意数量的页

面辅助对象组成。页面辅助对象包括元数据对象、数字签名对象、历史痕迹对象。

如图 8 所示，每个层对象包括一个或多个版面对象、任意数量的对象组和任意数量的层辅助对象。层辅助对象包括元数据对象、数字签名对象、历史痕迹对象。对象组包括任意数量的版面对象、任意数量的对象组和可选的数字签名对象。当对象组包括多个对象组时，其类似于资源管理器的文件夹包括多个文件夹的形式。

如图 9 所示，版面对象包括状态对象、文字对象、直线对象、曲线对象、圆弧对象、路径对象、渐变色对象、图像对象、流媒体对象、元数据对象、批注对象、语义信息对象、源文件对象、脚本对象、插件对象、二进制数据流对象、书签对象以及超链接对象。

其中，状态对象包括任意数量的字符集对象、字体对象、字号对象、文字颜色对象，光栅操作对象、背景色对象、线颜色对象、填充色对象、线型对象、线宽对象、线接头对象、画刷对象、阴影对象、阴影颜色对象、旋转对象、空心字对象、勾边字对象、透明对象和渲染模式对象。

在具体实施过程中，可以在上述通用文档模型基础上进一步增强或简化。如果在简化模型中省略了文档集对象，则文档库对象直接由文档对象组成；如果在简化模型中省略了层对象，则页面对象直接由版面对象组成。

可以理解，最简化的通用文档模型是仅包含文档对象、页面对象和版面对象。其中版面对象仅包括文字对象、直线对象和图像对象。完整模型和最简化模型之间的各种中间模型都属于本优选实施例的变形。

通用安全模型：

为了满足各种应用对文档安全性的需求，还需要定义一种通用的文档安全模型，以解决由于现有软件的文档安全功能不够强，或者是安全

管理机制与文档处理模块脱节所导致的安全缝隙。根据本发明一优选实施例，通用文档安全模型包括:

1. 在文档库中设置若干角色，角色对象是文档库对象的子对象。

2. 可以设置任意角色对任意对象（例如文档库、文档集、文档、页、层、对象组、版面对象等）的访问权限。如果设置了某角色对某对象的访问权限，则该权限适用于该对象的所有子对象。

3. 文档库系统实现的访问权限包括是否可读、是否可写、是否可再授权、是否可收回授权及其排列组合。也可以定义其他需要由应用软件来配合实现的权限，如不可打印。

4. 角色可以对任意对象进行签名。签名范围包括该对象的子对象，以及引用到的对象。

5. 创建角色对象的指令的执行结果是向应用软件返回一个密钥，作为应用软件以该角色身份登录的依据，该密钥通常是PKI的私钥，由应用软件保管，该密钥也可以是登录口令。

6. 当应用软件以某一角色身份登录时，通常采用“挑战－应答”机制，即文档库系统用保存的角色公钥加密一块随机数据发给应用软件，应用软件解密后返回给文档库系统，如果解密正确，则表明应用软件确实拥有该角色对应的私钥。“挑战－应答”机制也可以用以下方式实现，文档库系统将一块随机数据发给应用软件，应用软件用私钥加密后返回给文档库系统，文档库系统用保存的角色的公钥解密，如果解密正确，则表明应用软件确实拥有该角色对应的私钥。为保险起见，该认证过程可能会重复几次。采用“挑战－应答”机制可以更好地保护私钥的安全性。如果角色的密钥是登录口令，则需要用户输入正确的登录口令。

7. 应用软件可以同时登录多个角色，此时拥有的权限是各角色权限的并集。

在具体实施过程中，可以在上述通用安全模型基础上进一步增强或简化。针对上述安全模型的任何简化模型都是本实施例的变形。

接口层的具体实现

所述接口层的统一接口标准可根据通用文档模型、通用安全模型和常用的文档操作而定义，用于发送对通用文档模型中各对象进行操作的指令。所述的对通用文档模型中各对象进行操作的指令符合接口标准，各种应用软件可以通过接口层发出标准指令。

现在介绍接口标准的实现方式。接口标准的实现可以是上接口单元按照预先定义的标准格式生成命令串，例如“<UOML_INSERT (OBJ=PAGE, PARENT=123.456.789, POS=3) /> ”，将该命令串发送给下接口单元，并从下接口单元接收文档库系统对该命令的执行结果或其它反馈信息；或者，接口标准的实现是下接口单元提供一些具有标准名称和参数的接口函数，例如：“BOOL UOI_InsertPage (UOI_Doc *pDoc, int nPage)”，上接口单元调用这些标准函数，调用操作本身就代表上接口单元发出了标准指令；或者是上述方法的组合。

接口标准采用“操作动作+操作对象”的方式来实现便于学习和理解，也便于保持接口标准的稳定性。例如，对 20 种不同对象进行 10 种操作，可以定义 20x10=200 种指令，也可以定义 20 种对象和 10 种动作，但显然后一种方式大大减轻了记忆的负担，而且今后在对接口标准进行扩充时，增加一个对象或动作也很简单。所述操作对象为通用文档模型所包含的对象。

例如，定义以下 7 种操作动作：

打开：用于创建或打开文档库；

关闭：用于关闭会话句柄、关闭文档库；

获取：用于获取对象列表、对象相关属性和数据；

设置：用于设置/修改对象数据；

插入：插入指定对象或数据；

删除：用于删除对象的某个子对象；

检索查询：用于根据定义条件在文档中找到符合条件的内容，这些条件既可以是准确的信息，也可以是不准确的信息，即模糊查找。

定义如下对象：文档库、文档集、文档、页、层、对象组、文字、图像、图形、路径（由一组顺序图形连接组成，可以是闭合也可以不闭合的）、源文件、脚本、插件、音频、视频、角色等。

对象还包括下列状态对象：背景色、线的颜色、填充色、线型、线宽、ROP、画刷、阴影、阴影颜色、字符高、字符宽、旋转、透明、渲染模式等。

可以理解，在采用“操作动作+操作对象”方式实现接口标准时，不能自动理解为每一个对象和每一个动作的所有组合都一定能构成有实际意义的操作指令，一些组合是没有意义的。

还可以用非“操作动作＋操作对象”的函数方式来定义接口标准，例如对每一个对象的每一种操作都定义一个接口函数，这样各种操作指令就是上接口单元以调用下接口单元的接口函数来发送给文档库系统。

还可以封装各个对象类，如文档库类，把该对象可以进行的操作定义成该类的方法。

特别地，如果在接口标准中定义了获取版面位图的指令，将对保障版面一致性和文档互操作性起到非常关键的作用。

在检索查询指令中，除了常规的关键词检索外，还可以提供更加丰富的检索手段。在常规的搜索技术中，搜索是和文档处理分离的，搜索程序只能从文档中提取纯文本信息，而无法获取更多信息，只能基于文本信息检索。但在本发明中，检索查询功能是集成在文档处理的核心层，

即文档库系统，这样就可以更充分地利用文档中蕴含的信息来提供更为强大的检索手段，如:

1. 基于字体信息的检索，如检索黑体字的“书生”，Times New Roman字体的“Sursen”

2. 基于字号信息的检索，如检索三号字的“书生”，20 磅以上的“Sursen”，长字（即字高超过字宽）的“文档库”

3. 基于颜色的检索，如检索红色的“书生”，蓝色的“Sursen”

4. 基于版面位置的检索，如检索位于页面上半部分的“书生”，位于页脚的“Sursen”

5. 基于特殊修饰效果的检索，如检索斜体字的“书生”，顺时针旋转30 度至 90 度之间的“Sursen”，空心字的“SEP”，勾边字的“文档库”

6. 根据类似的思路，还可以进一步提供其它类型的检索，如检索反白（黑底白字）的“书生”，压图的“Sursen”等

7. 可以检索多个版面对象的组合，如“书生”距离“Sursen”不超过 5厘米

8. 上述检索条件的任意组合

以下是用“操作动作+操作对象”的方式实现接口标准的一个实施例，在该实施例中，接口称为非结构操作标记语言（UOML）,是用可扩展标记语言（XML）描述的一系列的命令。每个动作都对应一个 XML 元素，每个对象也都对应一个 XML 元素;描述一个命令时，将描述对象的 XML元素作为描述动作的 XML 元素的子元素，即可生成包括动作+对象的字符串。上接口单元将该字符串发送给下接口单元，就将相应的操作指令发送给了文档库系统。文档库系统执行这些命令后，下接口单元将执行结果也生成一个符合 UOML 格式的字符串，返回给上接口单元，使应用软件能够知晓操作执行结果。

所有执行结果都由 UOML_RET 表示，参见图 10，其定义如下：

属性：

SUCCESS:值为真（true）时表明操作成功，否则失败。

子元素：

ERR_INFO: 可选，仅当操作失败时出现，描述了相应的错误信息。

其它子元素：根据具体命令确定，可参考各命令说明。

UOML 动作包括：

1 UOML_OPEN 创建或打开文档库，参见图 11。

1.1 属性

1.1.1 create: 为 true 时是创建，否则是打开已有文档库。

1.2 子元素：

1.2.1 path:文档库路径。可以是磁盘文件名，也可以是 URL，或者是内存指针，或者是网络路径，或者是文档库的逻辑名称，或者其它能够指定文档库的表示方法。可以用不同特征的字符串区分上述各种情况，即不用改变命令格式，只要给字符串设置不同特征，就可以用不同的方法指定文档库。例如，磁盘文件名采用设备名称（如盘符）和":"开头（如"C:"、 "D:"），而且紧跟着":"不会是"//"，也不会是又一个":"；URL 采用协议名称和"://"开头（如"http://"）；内存指针用"MEM::"开头，后面是指针的字符串表示方式，例如"MEM::1234:5678"；网络路径是"\\"开头，后面是服务器名，以及服务器上的路径，如"\\server\abc\def.sep"；文档库的逻辑名称可以用"*"开头，如"*MyDocBase1"。在下接口单元解析时，如果第一个字母是"*"就表明该字符串代表文档库的逻辑名称；否则如果头两个字母是"\\"就表明该字符串代表网络路径；否则如果头五个字母是"MEM::"就表明该字符串代表内存指针；否则寻找字符串的第一个":"，如果该":"后面是"//" 该就表明字符串代表 URL，否则就代表本

地设备上的文件。对于打开服务器上的文档库的情形，可以设立一个专门的 URL 协议来区分，例如用“Docbase://myserver/mydoc2”指明打开服务器 myserver 上运行的文档库系统服务器系统所管理的 mydoc2 文档库。总之，只要能给字符串设置不同特征，就可以用不同的方式来指定文档库。根据上述说明，还可以定义各种不同的字符串特征；该方式不仅能应用于指定文档库路径，还能应用于其它场合，特别是用来指定特定资源位置的应用场合。在很多情况下，希望能够用一种新方式来指定相关资源，但又不能或不希望改变现有的协议或函数，这时就可以通过在字符串中设置不同特征的方式来指定，这种方法具有最好的通用性，这是因为，无论何种协议或函数，只要支持磁盘文件名或 URL，就支持字符串。

1.3 返回值:

如果成功，则在 UOML_RET 中包含一个“handle”子元素，记录句柄

2 关闭(UOML_CLOSE)，参见图 12。

2.1 属性：无。

2.2 子元素:

2.2.1 handle: 对象句柄，是一个字符串表示的对象的引用指针。

2.2.2 db_handle: 文档库句柄，字符串表示的文档库的引用指针。

2.3 返回值：无返回值。

3 UOML_GET 获取，参见图 13。

3.1 属性

3.1.1 usage: 用途，为”GetHandle”（获取指定对象句柄）、”GetObj”（获取指定对象数据）、”GetPageBmp”（获取版面位图）中的一个。

3.2 子元素

3.2.1 parent: 父对象句柄，usage 属性为”GetHandle”时使用。

3.2.2 pos: 位置顺序号，usage 属性为"GetHandle"时使用。

3.2.3 handle: 指定对象的句柄，当 usage 属性为"GetObj"时使用。

3.2.4 page: 需要显示的页面的句柄，当 usage 属性为"GetPageBmp"时使用。

3.2.5 input: 描述了对输入页面的约束，其中可以指定显示一层或者多层的内容（可以显示的层一定是当前角色有权限访问的层）；也可以通过指定 Clip 区域来指定显示区域的大小。当 usage 属性为"GetPageBmp"时使用。

3.2.6 output: 描述了版面位图的输出方式，当 usage 属性为"GetPageBmp"时使用。

3.3 返回值:

3.3.1 当 usage 属性为"GetHandle"时，执行成功时在 UOML_RET 中包含一个"handle"子元素，记录 parent 下第 pos 个子对象的句柄。

3.3.2 当 usage 属性为"GetObj"时，执行成功时在 UOML_RET 中包含一个"xobj"子元素，含有 handle 对象的数据的 xml 表示。

3.3.3 当 usage 属性为"GetPageBmp"时，执行成功时在 output 指定位置输出版面位图。

4 UOML_SET 设置，参见图 14。

4.1 属性: 无。

4.2 子元素:

4.2.1 Handle: 设置对象的句柄。

4.2.2 xobj: 对象的描述。

4.3 返回值: 无返回值。

5 UOML_INSERT 插入，参见图 15

5.1 属性: 无。

5.2 子元素:

5.2.1 parent: 父对象句柄。

5.2.2 xobj: 对象的描述。

5.2.3 pos: 插入位置。

5.3 返回值:如果执行成功,则将xobj参数表示的对象,插入到parent中成为其第pos个子对象,并在UOML_RET中包含一个"handle"子元素,表示新插入对象的句柄。

6 UOML_DELETE 删除,参见图16

6.1 属性: 无。

6.2 子元素:

6.2.1 handle: 需要删除的对象的句柄。

6.3 返回值: 无返回值。

7 UOML_QUERY 检索查询,参见图17

7.1 属性: 无。

7.2 子元素:

7.2.1 handle: 需要查询的文档库句柄。

7.2.2 condition: 查询条件。

7.3 返回值: 如果成功,在UOML_RET中包含一个"handle"子元素代表查询结果的句柄,一个"number"子元素代表查询结果的数量,可以用UOML_GET来获取每一个查询结果。

UOML对象包括:

文档库(UOML_DOCBASE)、文档集(UOML_DOCSET)、文档(UOML_DOC)、页(UOML_PAGE)、层(UOML_LAYER)、对象组(UOML_OBJGROUP)、文字(UOML_TEXT)、图像(UOML_IMAGE)、直线(UOML_LINE)、曲线(UOML_BEIZER)、圆弧(UOML_ARC)、路径

（UOML_PATH）、源文件(UOML_SRCFILE)、背景色（UOML_BACKCOLOR）、前景颜色(UOML_COLOR)、ROP(UOML_ROP)、字符尺寸(UOML_CHARSIZE)、字体(UOML_TYPEFACE)。

下文以 UOML_DOC、UOML_TEXT 和 UOML_CHARSIZE 为例说明其定义方式:

1 UOML_DOC

1.1 属性: 无。

1.2 子元素:

1.2.1 metadata: 元数据。

1.2.2 pageset: 各页面。

1.2.3 fontinfo: 嵌入字库。

1.2.4 navigation: 导航信息。

1.2.5 thread: 导读信息。

1.2.6 minipage: 微缩版面。

1.2.7 signiture: 数字签名。

1.2.8 sharesource: 共享资源。

2 UOML_TEXT

2.1 属性:

2.1.1 Encoding: 文字编码方式。

2.2 子元素:

2.2.1 TextData: 文字内容。

2.2.2 CharSpacingList: 对非等间距文字的字间距列表。

2.2.3 StartPos: 起点位置。

3 UOML_CHARSIZE

3.1 属性:

3.1.1 width: 字符宽度。

3.1.2 height: 字符高度。

3.2 子元素: 无。

以此类推，可以用同样的方法来描述所有的 UOML 对象。当应用软件对文档库进行操作时，由上述 UOML 动作与 UOML 对象依照 XML 语法生成相应的 UOML 命令，将该 UOML 命令发给文档库系统，即代表向文档库系统发出了相应操作指令。

例如,对创建文档库操作,可以用以下命令来完成:

```
<UOML_OPEN create="true">
  <path val="f:\\data\\docbase1.sep"/>
</UOML_OPEN>
```

对创建文档集操作,可以用以下命令来完成:

```
<UOML_INSERT >
  <parent val= "123.456.789"/>
  <pos val="1"/>
  <xobj>
    <docset/>
</xobj>
</UOML_INSERT>
```

需要说明的是，虽然 UOML 是用 XML 定义的，但为了显得更加简洁，在前面省略了类似“<?xml version="1.0" encoding="UTF-8"?>”以及“xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"”之类的常规 XML 格式，只要是熟悉 XML 语法的实施者都可以在实施过程中自行添加。

也可以不用 XML 方式定义命令串，例如改用类似 PostScript 那样的

方式，这样上例变成:

1, "f:\\data\\docbase1.sep", /Open

/docset, 1, “123.456.789” , /Insert

根据同样的思路，还可以定义出其它类型的命令串格式，甚至还可以不用文本方式，而用二进制方式来定义命令串。

现在介绍对每一个对象的每一个操作都用一个命令来表示的方式的一个具体实例，在本实例中，用“UOML_INSERT_DOCSET”来表示插入一个文档集，用“UOML_INSERT_PAGE”来表示插入一页，以这样的方式来定义每个命令:

UOML_INSERT_DOCSET 在文档库中创建一个文档集

属性: 无

子元素:

parent: 文档库句柄

pos: 插入位置

返回值: 如果执行成功，则在 UOML_RET 中包含一个”handle”子元素，表示新插入文档集的句柄

这样上例就变为:

<UOML_INSERT_DOCSET >

<parent val="123.456.789"/>

<pos val="1"/>

</UOML_INSERT_DOCSET >

用这种方法定义命令格式需要对每个对象的每种合法操作都单独定义一条命令，缺点是比较繁琐。

现在介绍用函数调用的方式来实现接口标准的实例，在该实例中，通过上接口调用下接口的接口函数的方式来发送操作指令给文档库系

统。以下以 C++语言为例说明，该实例称为 UOI。在该实例中，定义了 UOI_Object 作为所有对象的基类，为每个动作定义了一个函数，该函数的参数可以是对基类的指针或引用，这样该函数就可适用于所有对象。

先定义一个 UOI 返回值结构:

```
struct UOI_Ret {
    BOOL m_bSuccess;
    CString     m_ErrInfo;
};
```

定义所有 UOI 对象的基础类:

```
class UOI_Object {
public:
    enum Type {
        TYPE_DOCBASE,
        TYPE_DOCSET,
        TYPE_DOC,
TYPE_PAGE,
TYPE_LAYER,
TYPE_TEXT,
TYPE_CHARSIZE,
        ……
    };
    Type    m_Type;

    UOI_Object();
virtual ~ UOI_Object();
static UOI_Object Create(Type objType);
};
```

然后定义如下几个 UOI 函数,与“操作动作 + 操作对象”方式实例中的几个 UOML 动作相对应:

UOI_RET UOI_Open(char *path, BOOL bCreate, HANDLE *pHandle);

UOI_RET UOI_Close(HANDLE handle, HANDLE db_handle);

UOI_RET UOI_GetHandle(HANDLE hParent, int nPos, HANDLE *pHandle);

UOI_RET UOI_GetObjType(HANDLE handle, UOI_Object ::Type *pType);

UOI_RET UOI_GetObj(HANDLE handle, UOI_Object *pObj);

UOI_RET UOI_GetPageBmp(HANDLE hPage, RECT rect, void *pBuf);

UOI_RET UOI_SetObj(HANDLE handle, UOI_Object *pObj);

UOI_RET UOI_Insert(HANDLE hParent, int nPos, UOI_Object *pObj, HANDLE *pHandle = NULL);

UOI_RET UOI_Delete(HANDLE handle);

UOI_RET UOI_Query(HANDLE hDocbase, const char *strCondition, HANDLE *phResult);

然后定义各 UOI 对象,依然以 UOI_Doc、UOI_Text 和 UOML_CharSize 为例说明:

```
class UOI_Doc : public UOI_Object {
public:
    UOI_MetaData    m_MetaData;
    int             m_nPages;
    UOI_Page        **m_pPages;
int             m_nFonts;
UOI_Font        **m_pFonts;
```

```
UOI_Navigation  m_Navigation ;
UOI_Thread      m_Thread ;
UOI_MiniPage    *m_pMiniPages ;
UOI_Signature   m_Signature ;
int             m_nShared ;
UOI_Obj         *m_pShared;

UOI_Doc();
virtual ~UOI_Doc() ;
};

class UOI_Text : public UOI_Object {
public:
   enum Encoding {
      ENCODE_ASCII,
      ENCODE_GB13000,
      ENCODE_UNICODE,
      ......
   } ;
   Encoding  m_Encoding;
   char      *m_pText ;
   Point     m_Start ;
   int       *m_CharSpace ;

UOI_Text();
virtual ~ UOI_Text();
};
class UOI_CharSize : public UOI_Object {
```

```
public :
    int m_Width ;
    int m_Height ;

    UOI_CharSize();
    virtual ~UOI_CharSize();
};
```

以下说明 UOI 的使用方法。首先是创建文档库操作:

```
    ret = UOI_Open("f:\\data\\docbase1.sep", TRUE, &hDocBase);
```

然后是构建一个创建新对象的函数:

```
HANDLE InsertNewObj(HANDLE hParent, int nPos,
UOI_Object ::Type type)
{
UOI_Ret ret;
HADNLEhandle ;
UOI_Obj *pNewObj = UOI_Obj::Create(type);
if (pNewObj == NULL)
    return NULL;
ret = UOI_Insert(hParent, nPos, pNewObj, &handle) ;
delete pNewObj ;
return ret.m_bSuccess ? handle : NULL;
    }
```

然后是直接获取对象的函数:

```
UOI_Obj *GetObj(HANDLE handle)
{
UOI_Ret ret;
UOI_Object ::Type       type;
UOI_Obj *pObj;
```

```
ret = UOI_GetObjType(handle, &type);
if ( !ret. m_bSuccess )
    return NULL;
pObj = UOI_Obj::Create(type);
if (pObj == NULL)
    return NULL;
ret = UOI_GetObj(handle, pObj);
if ( !ret. m_bSuccess ) {
    delete pObj;
    return NULL;
}
return pObj;
}
```

在对每一个对象的每一种操作都定义一个接口函数的方式中，插入文档集的操作指令就是上接口以下列方式调用下接口的接口函数来发送给文档库系统的:

```
UOI_InsertDocset(pDocbase, 0);
```

在封装各个对象类（如文档库类）的方式中，把该对象可以进行的操作定义成该类的方法，如:

```
class UOI_DocBase : public UOI_Obj
{
public:
/*!
 *   \brief        创建文档库
 *   \param       szPath: 文档库全路径
 *   \param       bOverride:是否覆盖原文件
 *   \return     UOI_DocBase 对象
```

```
 */
  BOOL Create(const char *szPath, bool bOverride = false);
/*!
 *  \brief        打开文档库
 *  \param      szPath: 文档库全路径
 *  \return     UOI_DocBase 对象
 */
  BOOL Open(const char *szPath);
 /*!
 *  \brief          关闭文档库
 *  \param      无
 *  \return      无
 */
  void Close();
 /*!
 *  \brief          获取角色列表
 *  \param      无
 *  \return       UOI_RoleList 对象
 *  \sa           UOI_RoleList
 */
  UOI_RoleList GetRoleList();
/*!
 *  \brief          存储文档库
 *  \param      szPath: 存储文档库全路径
 *  \return       无
 */
```

```
    void Save(char *szPath = 0);

/*!
 *   \brief          插入文档集
 *   \param      nPos:插入文档集的位置
 *   \return       UOI_DocSet 对象
 *   \sa           UOI_DocSet
 */
  UOI_DocSet InsertDocSet(int nPos);
 /*!
 *   \brief          获取指定索引的文档集
 *   \param      nIndex: 文档列表的索引号
 *   \return       UOI_DocSet 对象
 *   \sa           UOI_DocSet
 */
  UOI_DocSet GetDocSet(int nIndex);
 /*!
 *   \brief          获取文档集的总数
 *   \param      无
 *   \return       文档集个数
 */
  int GetDocSetCount();
 /*!
 *   \brief          设置文档库的名称
 *   \param      nLen:          文档库名称长度
 *   \param      szName:   文档库名称
```

```
*   \return        无
*/
  void SetName(int nLen, const char* szName);
/*!
*   \brief          获取文档库名称长度
*   \param       无
*   \return        长度
*/
  int GetNameLen();
/*!
*   \brief          获取文档库名称
*   \param       无
*   \return        文档库名称
*/
  const char* GetName();
/*!
*   \brief          获取文档库 id 长度
*   \param       无
*   \return        长度
*/
  int GetIDLen();
/*!
*   \brief          获取文档库 id
*   \param       无
*   \return        id
*/
```

```
    const char* GetID();
    //! 构造函数
    UOI_DocBase();
    //! 析构函数
    virtual ~UOI_DocBase();
};
```

这样插入文档集的操作指令就是上接口以下列方式调用下接口的接口函数来发送给文档库系统的:

```
pDocBase.InsertDocset(0);
```

还可以用同样的方法为 Java、C#、VB、Delphi 等各种编程语言开发的应用软件设计各种不同的接口标准。

只要在接口标准中不含有与特定的操作系统（如 WINDOWS、UNIX/LINUX、MAC OS、SYMBIAN）或特定的硬件平台（如 x86CPU、MIPS、POWER PC 等）相关连的特征，该接口标准就可以具有跨平台性，使得不同平台上运行的应用软件和文档库系统都可以统一使用同样的接口标准，特别是可以让一个平台上运行的应用软件可以调用另一个平台上运行的文档库系统来执行相应操作。例如，应用软件部署在客户端，使用的是 PC 机，Windows 操作系统，文档库系统部署在服务器端，使用的是大型机，Linux 操作系统，但应用软件依然可以像调用本地文档库系统一样调用服务器上的文档库系统来执行相应文档操作。

如果在接口标准中不含有与特定编程语言相关的特征，则该接口标准还能做到与编程语言无关。可以看出，用命令串的方式容易构造与平台无关、与编程语言无关的接口标准，更具有通用性。特别是用 XML 来构造命令串的话，由于目前在各种不同平台、不同编程语言都存在易于获得的 XML 生成解析工具，因此不仅该接口标准具有很好的跨平台

性和与编程语言无关性，也非常便于工程师开发上接口单元和下接口单元。

以上列举了多种接口标准的实现方法，按照类似的思路设计的更多种类的接口标准也包含在本发明的保护范围之内。

应该理解，可以在上述实例的基础上按同样的思路增加操作指令，也可以简化操作指令，特别是文档模型被简化时操作指令也会相应被简化。最简化情况下只有文档的创建、页面的创建、各版面对象的创建这几个操作指令。

文档操作处理

现在，参见图 1，继续描述依照本发明一优选实施例的文档处理系统的工作过程。

应用软件是包含符合接口标准的上接口单元的软件，例如 Office 软件、内容管理、资源采集等。任一应用软件在需要对文档进行操作时，依照前述方法将指令传递给文档库系统，文档库系统根据指令来完成具体操作过程。

文档库系统可以自由地存储、组织文档库数据，例如可以把一个文档库的文件全部都存储在一个磁盘文件中；可以一个文档对应一个磁盘文件，利用操作系统中的文件系统功能实现多文档组织；也可以一页对应一个磁盘文件；还可以完全抛开操作系统，在磁盘上留出一块空间后直接对磁道、扇区进行管理。对文档库数据的存储格式，可以用二进制格式保存，可以用 XML，还可以用二进制 XML。页面描述语言（定义页面上的文字、图形、图像等对象的方法）可以采用 PostScript，可以采用 PDF、SPD（书生公司使用的页面描述语言），当然也可以采用自定义的任何页面描述语言，只要其符合统一的接口标准。

例如，可以用 XML 来描述文档库数据，当文档模型是层次型的时

候，可以完全对照建立相应的 XML 树。执行创建操作时就在 XML 树中增加一个结点，执行删除操作就删掉相应结点，执行设置操作就设置相应结点的属性，执行获取操作就取出相应结点的属性并返回给应用软件，执行查询操作时就遍历相关结点查找。

以下是该实施例的进一步说明:

1. 用 XML 来描述每个对象。也就是说，为每个对象都建立了一个对应的 XML 树。有的对象属性比较简单，其对应的 XML 树就只有根结点，有的对象比较复杂，其对应的 XML 树还有子结点。具体描述方法可以参见前面用 XML 来定义操作对象的说明。

2. 当新建一个文档库时就新建一个根结点为文档库对象的 XML 文件。

3. 每当在文档库中插入一个对象时，如文字对象，就将该对象对应的 XML 树插入到插入位置的父结点（如层）之下。这样，文档库中的每个对象都在文档库为根结点的 XML 树中有一个对应的结点。

4. 当删除一个对象时，就删除该对象对应的结点，其下属所有子结点也都被删除。删除过程是从叶子结点开始自下而上遍历的。

5. 设置一个对象属性时，将该对象对应的结点的属性设置成该属性。如果该属性是用子结点表示的，则设置对应的子结点。

6. 获取一个对象属性时，访问该对象对应的结点，根据该结点的属性和子结点获得该对象的属性。

7. 获取一个对象的句柄时，返回该对象对应结点的 XML 路径。

8. 复制一个对象（如页面）到指定位置时，就将该对象对应的结点开始的整个子树都复制到目标位置对应的父结点（如文档）之下。如果是复制到另一个文档库中，则需要将该子树引用的对象（如嵌入字库）也一起复制过去。

9. 执行获取版面信息指令时，先生成一个指定位图格式的空白位图，其尺寸和指定区域相同，然后遍历指定页面的所有版面对象，凡是位于指定区域内（包括只有一部分在该区域内）的版面对象，都解释其含义，并在版面上相应体现。具体过程虽然比较复杂比较专业，但均属于现有 RIP 技术范畴，不在此赘述。

文档安全处理

在创建角色对象时，生成一对随机公私钥对（例如 512 位的 RSA 密钥），将公钥存储在角色对象中，将私钥返回给应用软件。

当应用软件登录时，随机生成一块（例如 128 字节）数据，用相应角色对象中的公钥加密该数据发给应用软件，应用软件解密后比较验证，如果正确则表明应用软件确实拥有该角色对应的私钥，登录成功。为保险起见，该认证过程可以重复三次，三次全部通过才算登录成功。

当对某一对象进行签名时，也就是对其对应的结点开始的子树进行签名。为了能够使签名不受具体物理存储方式的影响，需要先做一个正则化，使得逻辑上等效的变化（例如存储位置的改变导致相应指针的变化）不会影响签名有效性。该正则化的方法如下：

按深度优先遍历以目标对象为根节点的子树中的各个节点（即目标对象及其各个子对象），按照遍历顺序依次计算每个节点的正则结果并连接起来。

其中，对子树的某一节点计算正则结果的方法为：先计算该节点的子节点数的 HASH 值，然后再依次计算该节点类型及其各个属性的 HASH 值并按顺序连接在该节点的子节点数的 HASH 值的后面，再计算该连接结果的 HASH 值，得到该节点的正则结果。如果需要对子树中的某个节点引用的对象也一起做签名，则可以将该节点引用的对象也作为该节点的一个子节点来处理，方法同上。

正则化以后，再做HASH并用角色的私钥进行签名的处理可采用现有技术，这里不再赘述。

在上述正则化过程中，可以把计算一个节点正则结果的方法改成如下方案：将该节点的子节点数、类型及其各属性用分隔符隔开后按照顺序连接起来，计算该连接的结果的HASH值，得到该节点的正则结果。还可以把计算一个节点正则结果的方法改成如下方案：将该节点的子节点数、类型及其各属性的长度用分隔符隔开后按照顺序连接起来，再与子节点数、类型、各属性连接起来，即得到该节点的正则结果。总之，计算一个节点正则结果的方法可以采用以下各种方案中的任意一种：对树的某一节点，其子节点数、类型、各属性，子节点数／类型／各属性的长度（可选的），原值或经过特定变换（如HASH、压缩），按照预定顺序连接起来（直接连接或用分隔符隔开）。

上述预定顺序的意思是，子节点数长度、类型长度、各属性长度、子节点数、类型、各属性可以按任意顺序排列，只要是预定的顺序即可。

另外，在遍历子树中各个节点时，既可以采用深度优先遍历也可以采用宽度优先遍历。

不难给出上述方案的各种变化方式，如每个结点的子结点数用分隔符隔开后按照深度优先的顺序连接起来，再与各结点其它数据的正则结果连接起来。总之，只要对该子树中的所有结点的子结点数、类型和各属性，按照确定的方法排列在一起就属于本实施例的变化。

当对某一对象设置权限时，最简单的实现方式是简单记录各角色对该对象（及其子对象）的权限，并在今后各角色访问时加以比较，符合权限的则允许相应操作，否则报错返回。更好的实现方式是对相应数据加密，并用密钥来控制权限，如果该角色没有相应密钥就没有对应的权限，这种方式抗攻击能力要更强。具体方案为：

a) 对受保护的数据区域（通常为一个子树，对应某对象及其所有子对象），有一对对应的 PKI 密钥对，用其中的加密密钥对该数据区域进行加密。

b) 对具有读权限的角色，授予其解密密钥，该角色可以用该密钥解密该数据区域，从而正确读取这些数据。

c) 对具有写权限的角色，将授予其加密密钥，该角色可以将修改后的数据用该密钥加密，从而可以正确写入该区域的数据。

d) 鉴于 PKI 的加密／解密效率较低，为提高运行效率，也可以用对称密钥来对该数据区域加密，加密密钥用于对该对称密钥进行加密，解密密钥用于解密经过加密后的密钥数据，从而获得正确的对称密钥。为防止只有读权限的角色在获得对称密钥后用其修改数据，可以用加密密钥来对该数据区域进行数字签名，每次拥有写权限的角色修改该数据区域后都重新做一次签名，从而确保数据不会被没有写权限的角色篡改。

e) 当授予某一角色加密密钥或解密密钥时，可以用该角色的公钥对该密钥加密后存储，这样只有拥有该角色的私钥时才能取出该密钥

需要说明的是，本发明中所说明的文档安全技术，如基于角色的权限管理、角色的认证方式、多重角色登陆、对树结构的正则化技术、细粒度的权限管理单元、基于加密的权限设置等，都不仅适用于本发明所述的文档处理系统，还可以运用于更为广泛的其它应用场合。

对层的管理

在本发明中，为了使本文档处理系统能很好地模拟纸张的特性，提供了一种“只加不改”的技术方案。也就是说，每个应用软件都只在现有文档内容基础上添加新的内容，但不修改、不删除已有的内容，使文档的一个页面就象一张纸一样，可以由不同的人用不同的笔在纸上不断写

写画画，但谁都不能修改、删除已有内容。具体方法是每一个应用软件在编辑其它软件生成的文档时，都在现有文档基础上新增加一层，将本软件新编辑的内容都放到这一层中，不修改和删除前面各层的内容。这样，每个文档的每一层只由一个应用软件来管理和维护，其他应用软件不能对该层进行编辑。由于现有社会就是基于纸张来运转的，因此只要能符合纸张的特性就能满足现有应用的需求，具备足够的实用价值。

为了确保每一层内容在生成后没有被修改、删除，可以利用每一层的数字签名对象。数字签名可以是对本层内容进行签名，优选地，可以是对本层以及本层之前生成的所有层的内容一起签名。签名以后并不妨碍对文档做进一步的批注等编辑，只要新的内容是位于新建的层，没有修改破坏签名时存在的各层，签名依然是有效的，但签名者只对签名以前的内容负责，不对签名以后的内容负责。这是一个非常符合应用需求的技术方案，具有很大的实用价值。相比之下，现有的其它技术或者签名后不允许编辑，或者编辑后（尽管是"只加不改"的编辑）签名被破坏。

前述技术方案不允许修改文档中的已有内容，即使不考虑与纸张特性的兼容以及数字签名问题，需要修改的话也只能做版面级编辑，即对每个版面对象的编辑（增、删、改）都不会对其它版面对象产生影响（这是由于通用文档模型是基于可见部分为基础构建的，不包含大量不可见的、关于版面对象之间的关系，因此修改任何一个版面对象时，其它版面对象不会产生相应的调整，例如删掉一个字，就会在其位置留下空白，右边的文字不会自动左移）。如果用户需要对文档中的已有内容进行编辑，并且还希望能像在原来那样编辑的话，有一个技术方案可以很好地满足这个应用需求。该方案是当应用软件完成初始编辑时，除了新建一层存放当前编辑的内容外，还将源文件（按照应用软件自有的格式存储，记录了各对象之间完整关系的文件，例如.doc 文件）嵌入到文档中。当

下次需要进行继续编辑时，从文档中取出该源文件，并使用该源文件继续编辑。编辑完成后清除该软件所管理的那一层，重新生成该层的内容，并继续将新修改的源文件嵌入到文档中。

具体方法如下：

1.应用软件第一次处理该文档时，新建一层，将新编辑内容对应的版面对象插入到新建层中，同时用自身格式另存一份新编辑的内容（即源文件）。

2.在文档对象中新建一个源文件子对象，用来嵌入源文件（例如用二进制数据的方式整体嵌入），并记录是哪一层对应该源文件对象。

3.用同一应用软件再次编辑该文档时，从对应的源文件对象中取出对应的源文件。

4.使用该源文件继续编辑该层内容。由于该源文件是该应用软件自身的格式，可以按照该应用软件自身的功能继续对该层内容进行编辑。

5.再次编辑结束后，根据新编辑后的结果更新该层内容（例如用全部清除后全部重新生成的方式），同时将新修改后的源文件重新嵌入到文档对象中。

6.如此循环往复，就可以用原有应用软件按照原有方式对文档中的已有内容进行编辑。

采用上述技术方案，可以最大程度地实现文档的互操作性。在应用软件、文档都采用本发明技术时，在有足够安全权限的前提下，可以实现以下功能：

1. 对任何文档，用任何应用软件都可以正确打开、显示、打印。

2. 对任何文档，用任何应用软件都可以新添加任何内容，而且不会破坏文档已有签名。

3. 对任何文档，在不必考虑文档已有签名（没有签名或者虽有签名

但允许破坏）的前提下，用任何应用软件都可以对文档已有内容进行版面级编辑。

4. 对任何文档，使用文档已有内容的原始编辑软件可以对该内容进行正常编辑。

由此可见，通过本发明中对层的管理，对文档的管理、互操作、安全设置都带来极大的便利。

下面以A软件创建一个文档并且B软件对其进行编辑为例说明其工作过程。在本例中选用 UOI 作为接口标准:

1. A 软件发出指令,创建文档库 c:\sample\mydocbase.sep, 将其句柄存放在 hDocBase:

UOI_Open("c:\\sample\\mydocbase.sep", TRUE, &hDocBase);

2. A 软件发出指令,在文档库 hDocBase 中新建文档集, 将其句柄存放在 hDocSet:

hDocSet = InsertNewObj(hDocBase, 0, UOI_Obj:: TYPE_DOCSET); 在本实例中，该文档库中只有一个文档集，即第一个文档集;

3. A 软件发出指令,在文档集 hDocBase 中新建文档, 将其句柄存放在 hDoc:

hDoc = InsertNewObj(hDocSet, 0, UOI_Obj:: TYPE_DOC); 在本实例中，该文档集只有一个文档，即第一个文档;

4. A 软件发出指令,在文档 hDoc 中新建一页,版心大小是宽 w,高 h，将其句柄存放在 hPage:

UOI_Page page;

page.size.w = w;

page.size.h = h;

UOI_Insert(hDoc, 0, &page, &hPage); 在本实例中，该文档中只有一页，

即第一页;

5. A 软件发出指令,在页 hPage 中创建一层, 将其句柄存放在 hLayer:

hLayer = InertNewObj(hPage, 0, UOI_Obj::TYPE_LAYER); 在本实例中, 该页只有一层, 即第一层;

6. A 软件发出指令, 设置字号为 s:

UOI_CharSize charSize;

charSize.m_Width = charSize.m_Height = s;

UOI_Insert(hLayer, 0, &charSize); 在本实例中, 该层的第一个版面对象是字号对象;

7. A 软件发出指令,在坐标(x1,y1)位置插入文字串“书生意气挥斥方遒”:

UOI_Text text;

text.m_pText = Duplicate(“书生意气挥斥方遒”);

text.m_Encoding = UOI_Text:: ENCODE_GB13000;

text.m_Start.x = x1;

text.m_Start.y = y1;

UOI_Insert(hLayer, 1, &text); 在本实例中, 该层的第二个对象是文字对象;

8. A 软件发出指令,关闭文档库 hDocBase:

UOI_Close(hDocBase);

B 软件发出指令,打开文档库 c:\sample\mydocbase.sep, 将其句柄存放在 hDocBase:

UOI_Open(“c:\\sample\\mydocbase.sep”, FALSE, &hDocBase);

B 软件发出指令,获取文档库 hDocBase 第一个文档集的指针, 将其句柄存放在 hDocSet:

UOI_GetHandle(hDocBase, 0, &hDocSet);

9. B 软件发出指令,获取文档集 hDocSet 第一个文档的指针，将其句柄存放在 hDoc:

UOI_GetHandle(hDocSet, 0, &hDoc);

10. B 软件发出指令,获取文档 hDoc 第一页的指针,将其句柄存放在 hPage:

UOI_GetHandle(hDoc, 0, &hPage);

11. B 软件获取该页版面位图，用于显示该页

UOI_GetPageBmp(hPage, rect, buf);

12. B 软件发出指令,获取 hPage 第一层的指针，将其句柄存放在 hLayer:

UOI_GetHandle(hPage, 0, &hLayer);

13. B 软件发出指令,获取第一个版面对象的句柄 hObj:

UOI_GetHandle(hLayer, 0, &hObj);

14. B 软件发出指令,获取 hObj 的类型

UOI_GetObjType(hObj, &type);

15. B 软件发现这是一个字号对象，获取该对象

UOI_GetObj(hObj, &charSize);

16. B 软件将字高放大一倍:

charSize.m_Height *= 2;

UOI_SetObj(hObj, &charSize);

B 软件重新获取版面位图并显示,这时会发现屏幕上的“书生意气挥斥方遒”变成长体字了

下面，参照图 10 描述依照本发明的文档操作系统执行一操作的一个实例。在该实例中，应用软件通过统一的接口标准（例如 UOML 接口）请求对文档的操作。文档库系统可能会有不同厂商的不同型号，但是对

于应用开发厂商来说面向的都是同一个接口标准，因此都可以与之配套使用。Red Office、OCR、网页生成软件、乐谱编辑软件、书生阅读器、Office 编辑软件、其他阅读器等通过 UOML 接口指示文档库系统进行操作，文档库系统可以有多个，在图中显示为文档库系统 1、文档库系统 2 和文档库系统 3，文档库系统根据 UOML 发来的统一标准指令对符合通用文档模型的文档进行操作，例如创建、保存、显示、呈现文档。在本发明中，不同的应用软件可以同时或不同时调用同一个文档库系统，同一应用软件可以同时或不同时调用不同的文档库系统。

依照本发明，使得应用层和数据处理层分离，使得同一文档能在不同的应用软件之间通用，使不同应用软件之间具有良好的文档互操作性。

依照本发明，形成产业分工，减少重复开发，并更加专业、完备、正确；对文档的基本操作都在文档库系统中处理，各应用软件不必重复开发。而且由于文档库系统是由专业厂商开发，相关技术的专业性、完备性、正确性较有保障，而且应用软件厂商和用户可以选择做的最好的一家文档库系统厂商，从而保证处理效果的正确性和一致性。

依照本发明，提供多文档甚至海量文档的管理机制，使文档之间能够有效组织起来，便于检索、查询、保管，便于嵌入较强的信息安全机制。

依照本发明，提供更好的安全机制，可以设置多种角色，细粒度地设置每个角色的权限。其中细粒度是双重的，一方面可以对整个文档或文档的一个细微之处进行权限设置，另一方面可以设置种类非常多的权限，而不仅仅是传统的读／写／不可访问三级。

依照本发明，鼓励创新，合理竞争。形成合理的产业分工后，各文档库系统厂商和各应用软件厂商就会在领域展开竞争，而不会再出现

Microsoft Word 一样靠文档格式来垄断应用软件的情形发生。各文档库系统厂商也可以在标准之外增加新的功能以吸引用户，标准并不会对创新形成束缚。

依照本发明，便于优化性能，有更好的可移植性和可伸缩性。无论是什么平台，什么样的性能，都可以遵循同样的调用接口，使得在不改变接口标准的情况下可以不断优化性能，并移植到不同的平台。

以上所述仅为本发明的较佳实施例而已，并不用以限制本发明，凡在本发明的精神和原则之内所作的任何修改、等同替换和改进等，均应包含在本发明的保护范围之内。

权利要求书

1、一种文档处理系统，其特征在于，该系统包括应用层和文档库系统，所述应用层包括至少一个应用软件；

应用软件，用于向文档库系统发出包含动作和对象的、对文档进行操作的标准指令；

文档库系统，用于对所存储的文档数据中的所述对象，执行标准指令指示的动作。

2、根据权利要求 1 所述的系统，其特征在于，

应用软件包含上接口单元，用于发出包含动作和对象的标准指令；

文档库系统包含下接口单元，用于接收所述标准指令。

3、根据权利要求 1 所述的系统，其特征在于，所述动作包括以下至少其中一个：打开、关闭、获取、设置、插入、删除、检索查询。

4、根据权利要求 1 所述的系统，其特征在于，所述对象包括以下至少其中一个：文档库对象、文档集对象、文档对象、页对象、层对象、对象组、状态对象、文字对象、图像对象、图形对象、路径对象、源文件对象、脚本对象、插件对象、流媒体对象、链接对象、印章对象、二进制数据流对象、书签对象、批注对象、语义信息对象、元数据对象、角色对象、权限对象、数字签名对象、字库对象、导航信息对象、导读信息对象、微缩版面对象、索引信息对象、历史痕迹对象。

5、一种文档处理方法，其特征在于，该方法包括：

应用软件向文档库系统发出包含动作和对象的、对文档进行操作的标准指令；

文档库系统对所存储的文档数据中的所述对象，执行标准指令指示的动作。

6、根据权利要求 5 所述的方法，其特征在于，所述发出标准指令包括：分别定义各对象和各动作，确定应用软件的操作需求针对的对象和动作，根据所述对象和动作的定义生成对应的标准指令。

7、根据权利要求 6 所述的方法，其特征在于，所述标准指令为包含动作和对象的字符串。

8、根据权利要求 7 所述的方法，其特征在于，所述字符串为可扩展标记语言 XML。

9、根据权利要求 8 所述的方法，其特征在于，所述每个动作对应一个 XML 元素，所述每个对象对应一个 XML 元素；

所述生成标准指令包括：

生成与动作对应的第一 XML 元素，将与对象对应的 XML 元素作为所述第一 XML 元素的子元素。

10、根据权利要求 6 所述的方法，其特征在于，所述每个对象对应同一个基类派生的一种类型，所述每个动作对应一个接口函数，该接口函数的参数包括对基类的指针或引用；

所述发出标准指令包括：根据操作需求调用接口函数。

11、根据权利要求 10 所述的方法，其特征在于，所述接口函数是 C、C++、Java、C#、VB 或 Delphi 的函数。

12、根据权利要求 5 至 11 任一项所述的方法，其特征在于，所述动作包括以下至少其中一个：打开、关闭、获取、设置、插入、删除、检索查询。

13、根据权利要求 12 所述的方法，其特征在于，

所述打开动作包括以下参数：路径；

所述关闭动作包括以下参数：被关闭对象或其句柄；

所述获取动作包括以下参数：用途属性、父对象或其句柄；

所述设置动作包括以下参数：待设置对象或其句柄、对象描述；

所述插入动作包括以下参数：父对象或其句柄、对象描述、插入位置；

所述删除动作包括以下参数：待删除对象或其句柄；

所述检索查询动作包括以下参数：待查询对象或其句柄、查询条件。

14、根据权利要求 5 至 11 任一项所述的方法，其特征在于，所述对象包括以下至少其中一个：文档库对象、文档集对象、文档对象、页对象、层对象、对象组、状态对象、文字对象、图像对象、图形对象、路径对象、源文件对象、脚本对象、插件对象、流媒体对象、链接对象、印章对象、二进制数据流对象、书签对象、批注对象、语义信息对象、元数据对象、角色对象、权限对象、数字签名对象、字库对象、导航信息对象、导读信息对象、微缩版面对象、索引信息对象、历史痕迹对象。

15、根据权利要求 5 所述的方法，其特征在于，所述应用软件发出标准指令包括：预先定义对每个对象执行每种动作的标准指令，根据应用软件的操作需求确定对应的标准指令。

16、一种文档处理系统，其特征在于，该系统包括应用层、文档库系统和接口层，所述应用层包括至少一个应用软件；

应用软件，用于通过接口层，向文档库系统发出包含所述动作和对象的、对文档进行操作的标准指令；

文档库系统，用于通过接口层接收该标准指令，对所存储的文档数据中的对象执行标准指令指示的动作。

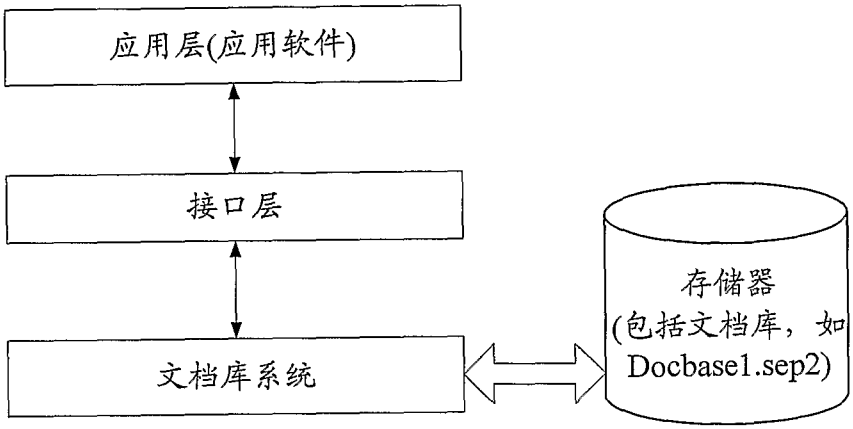

应用层(应用软件)
接口层
文档库系统
存储器
(包括文档库，如
Docbase1.sep2)

图 1

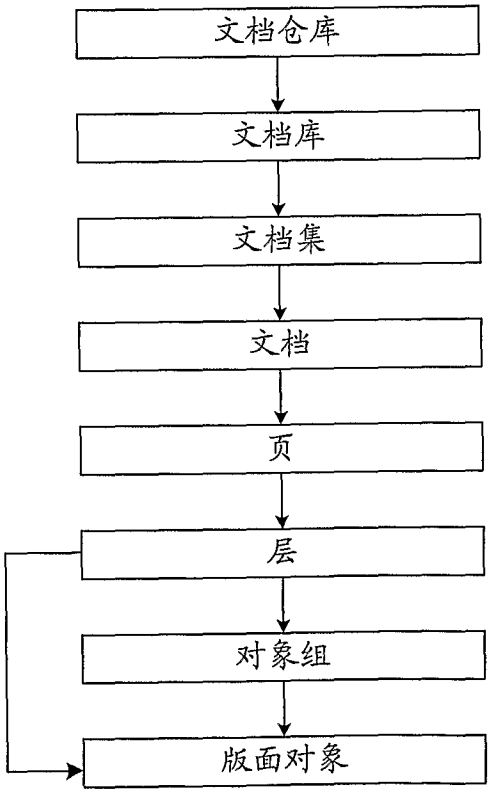

文档仓库
文档库
文档集
文档
页
层
对象组
版面对象

图 2

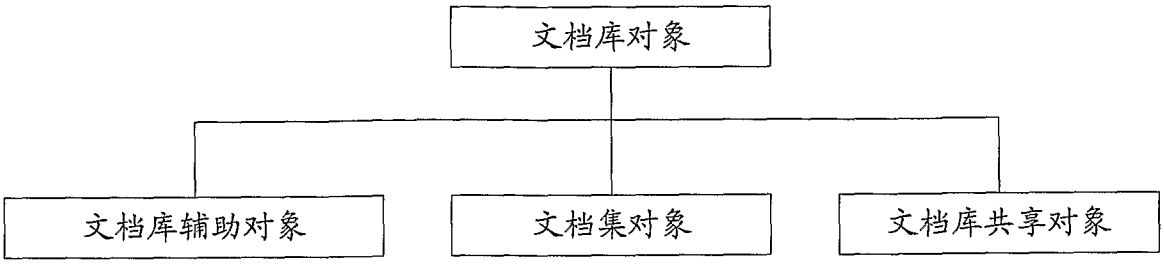

文档库对象
文档库辅助对象
文档集对象
文档库共享对象

图 3

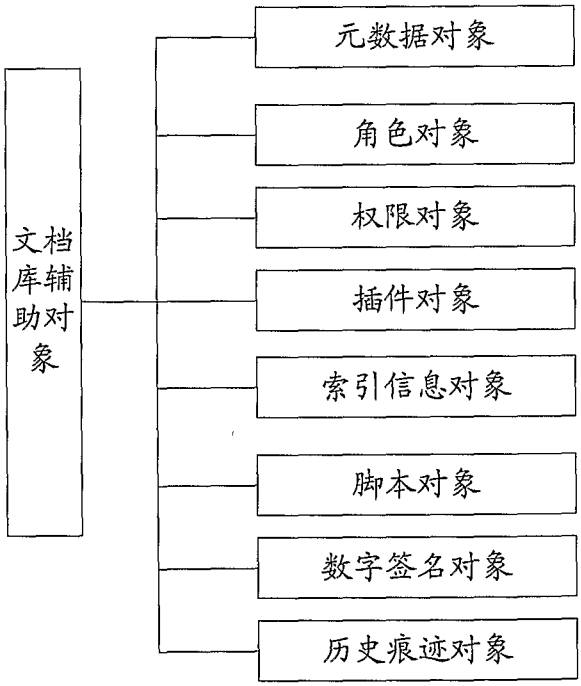

文档库辅助对象
元数据对象
角色对象
权限对象
插件对象
索引信息对象
脚本对象
数字签名对象
历史痕迹对象

图 4

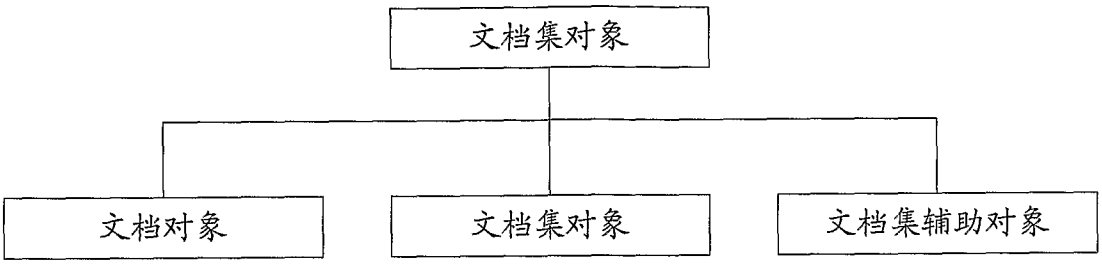

文档集对象
文档对象
文档集对象
文档集辅助对象

图 5

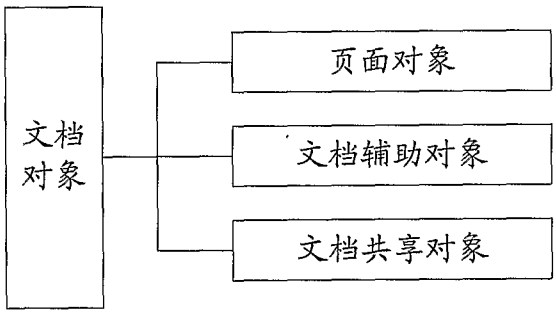

文档对象
页面对象
文档辅助对象
文档共享对象

图 6

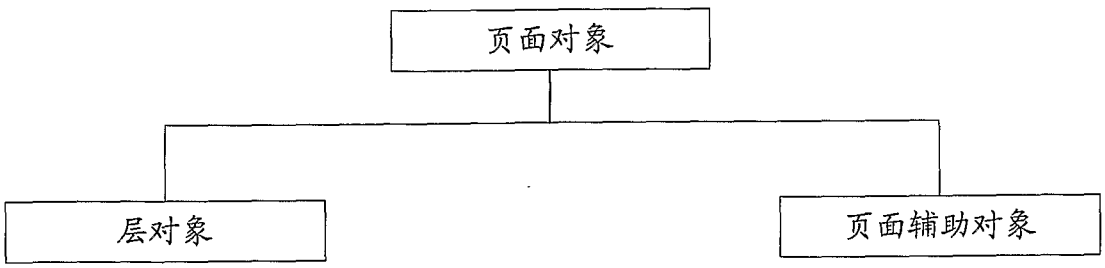

页面对象
层对象
页面辅助对象

图 7

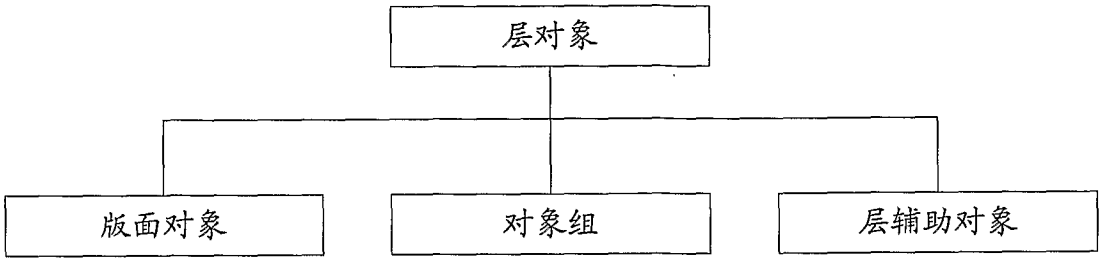

层对象
版面对象
对象组
层辅助对象

图 8

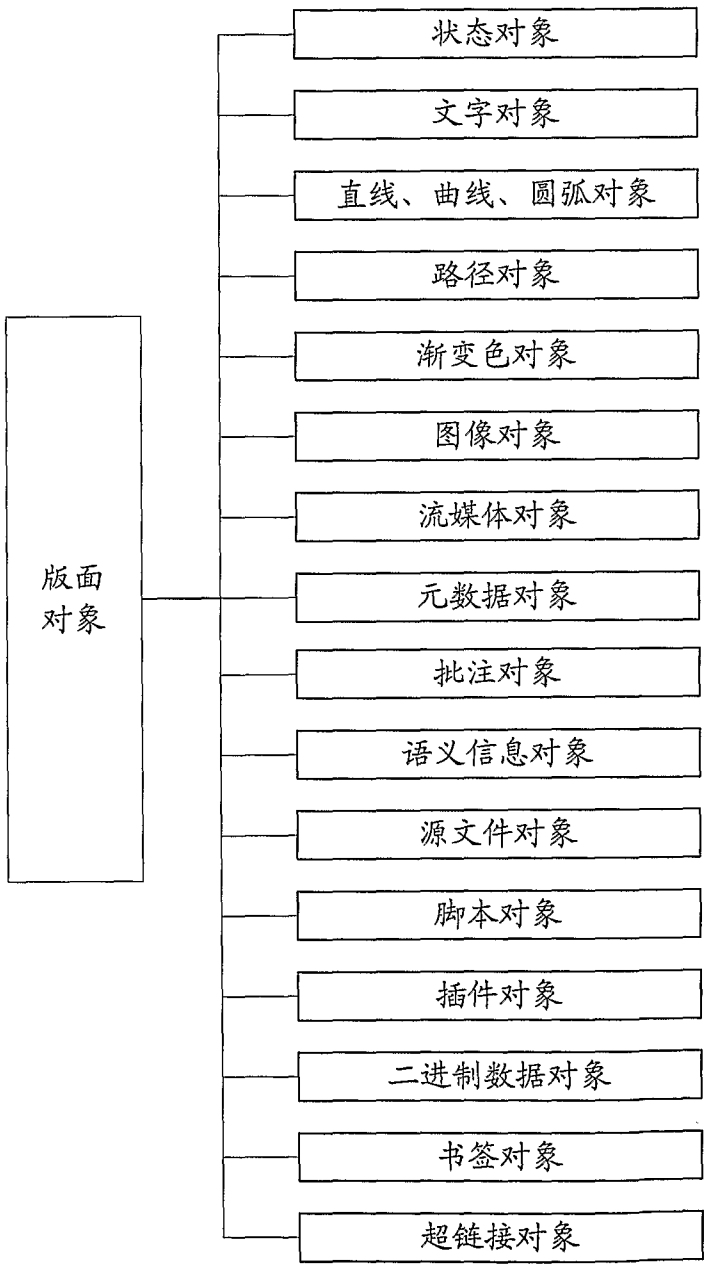

版面对象
状态对象
文字对象
直线、曲线、圆弧对象
路径对象
渐变色对象
图像对象
流媒体对象
元数据对象
批注对象
语义信息对象
源文件对象
脚本对象
插件对象
二进制数据对象
书签对象
超链接对象

图 9

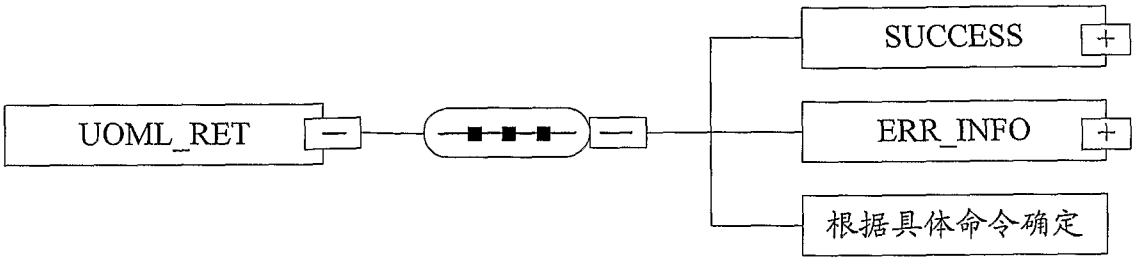

UOML_RET
SUCCESS
ERR_INFO
根据具体命令确定

图 10

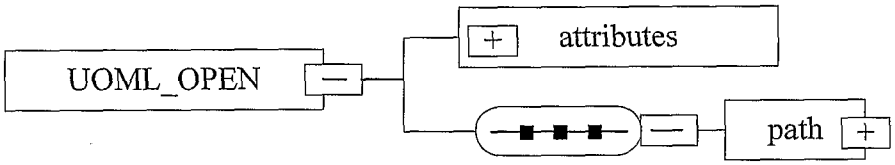

UOML_OPEN
attributes
path

图 11

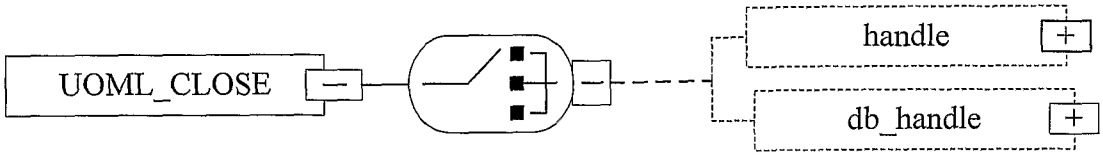

UOML_CLOSE
handle
db_handle

图 12

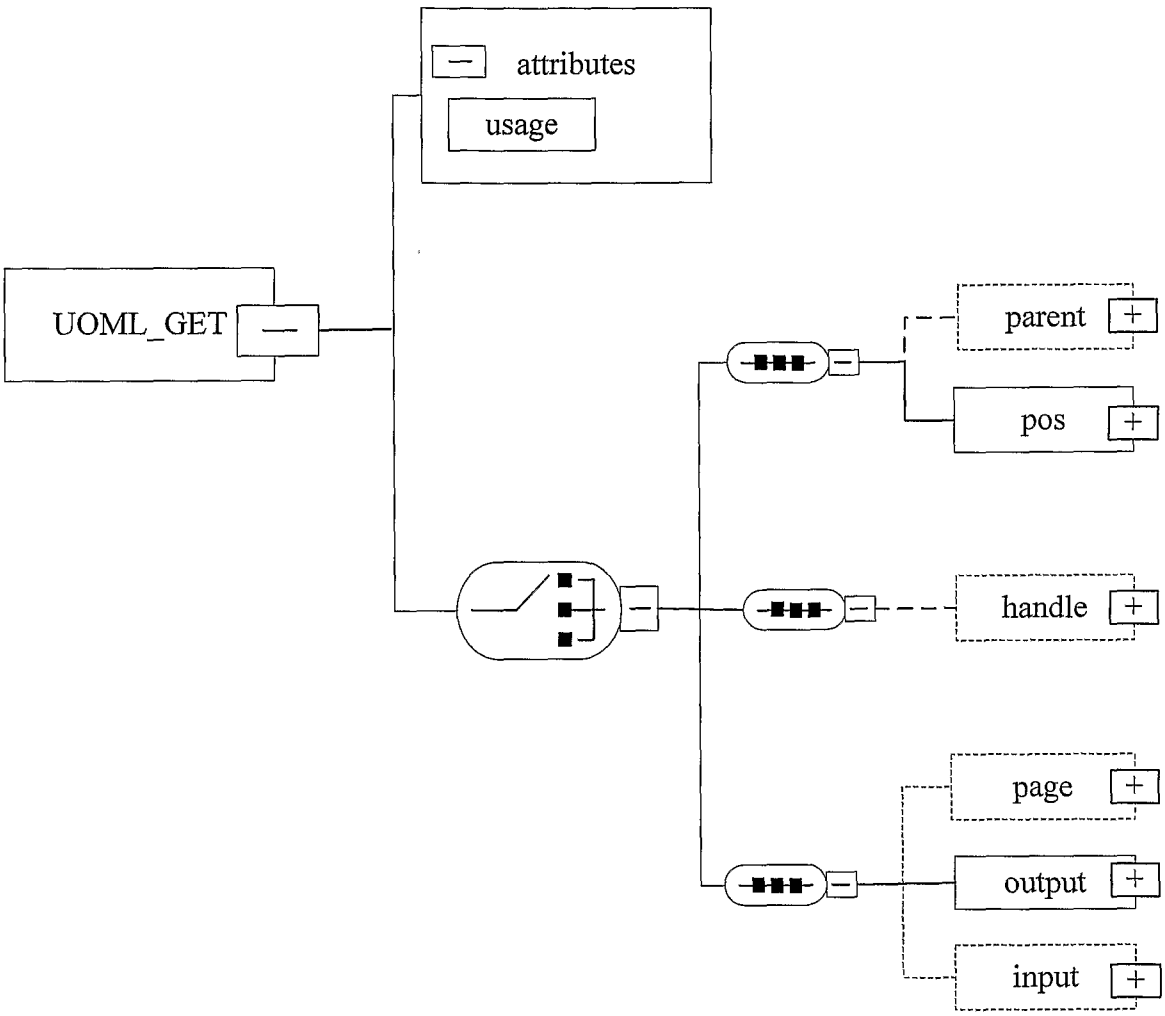

attributes
usage
UOML_GET
parent
pos
handle
page
output
input

图 13

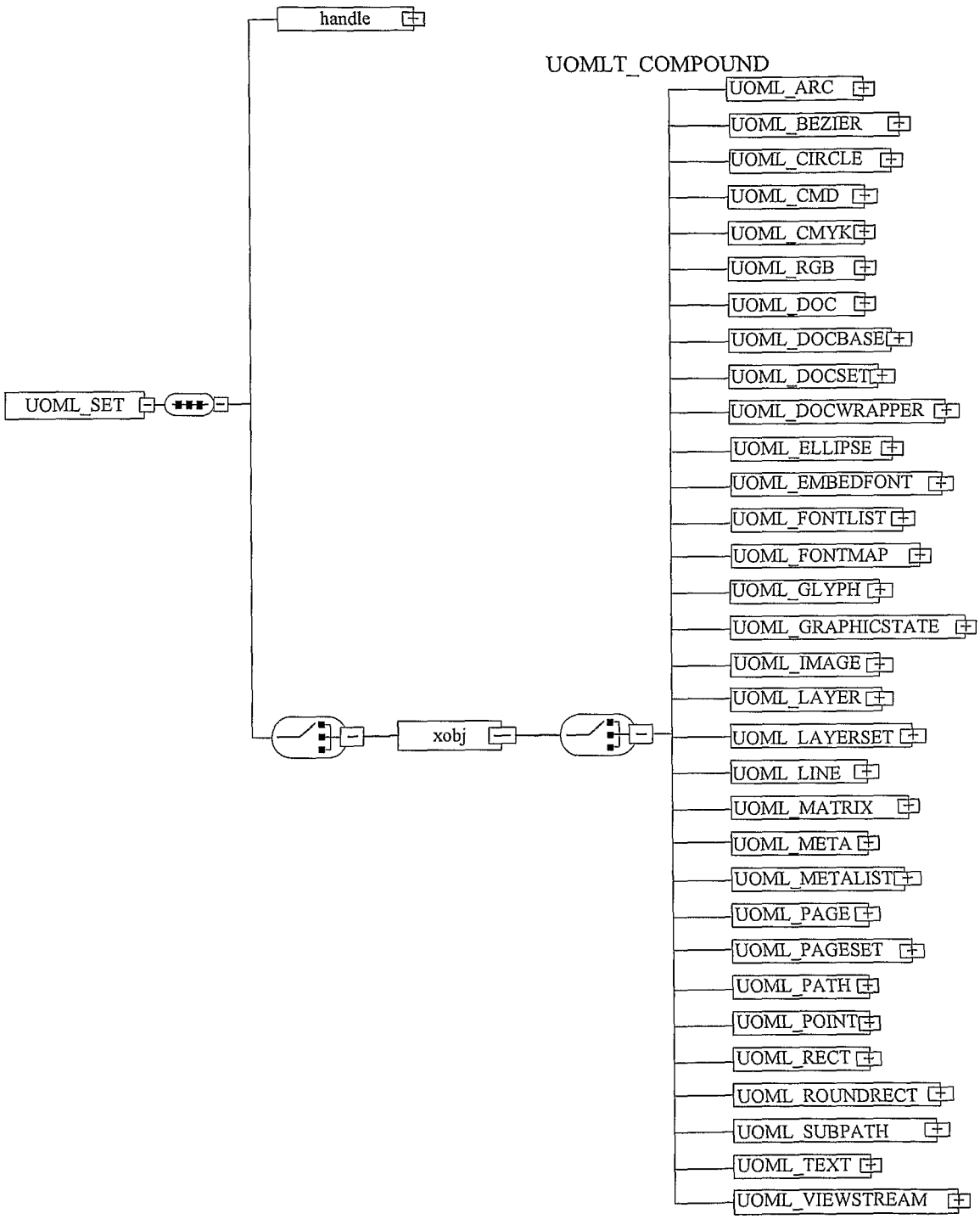

handle
UOML_SET
xobj
UOMLT_COMPOUND
UOML_ARC
UOML_BEZIER
UOML_CIRCLE
UOML_CMD
UOML_CMYK
UOML_RGB
UOML_DOC
UOML_DOCBASE
UOML_DOCSET
UOML_DOCWRAPPER
UOML_ELLIPSE
UOML_EMBEDFONT
UOML_FONTLIST
UOML_FONTMAP
UOML_GLYPH
UOML_GRAPHICSTATE
UOML_IMAGE
UOML_LAYER
UOML_LAYERSET
UOML_LINE
UOML_MATRIX
UOML_META
UOML_METALIST
UOML_PAGE
UOML_PAGESET
UOML_PATH
UOML_POINT
UOML_RECT
UOML_ROUNDRECT
UOML_SUBPATH
UOML_TEXT
UOML_VIEWSTREAM

图 14

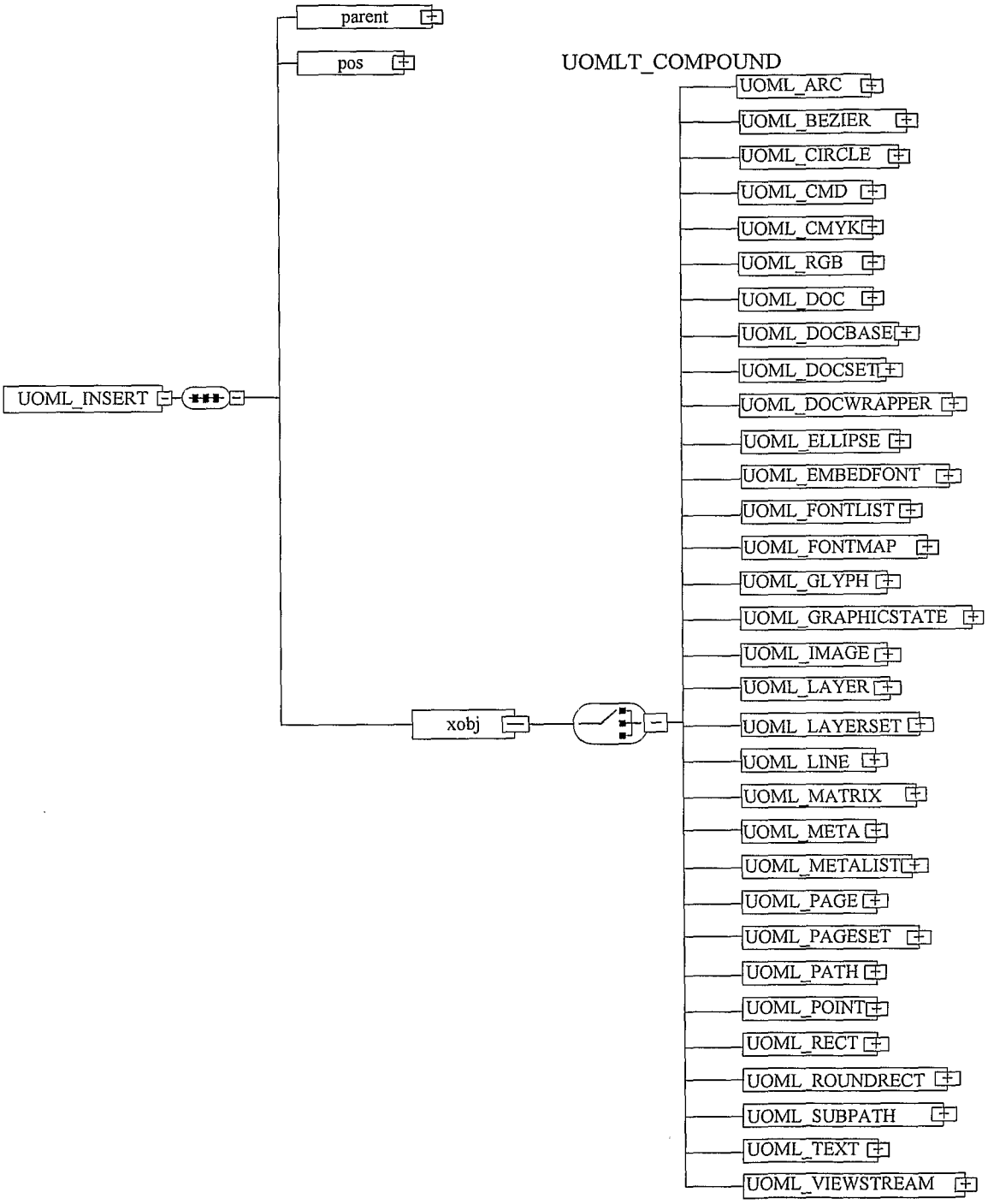

parent
pos
UOMLT_COMPOUND
UOML_ARC
UOML_BEZIER
UOML_CIRCLE
UOML_CMD
UOML_CMYK
UOML_RGB
UOML_DOC
UOML_DOCBASE
UOML_DOCSET
UOML_DOCWRAPPER
UOML_ELLIPSE
UOML_EMBEDFONT
UOML_FONTLIST
UOML_FONTMAP
UOML_GLYPH
UOML_GRAPHICSTATE
UOML_IMAGE
UOML_LAYER
UOML_INSERT
xobj
UOML_LAYERSET
UOML_LINE
UOML_MATRIX
UOML_META
UOML_METALIST
UOML_PAGE
UOML_PAGESET
UOML_PATH
UOML_POINT
UOML_RECT
UOML_ROUNDRECT
UOML_SUBPATH
UOML_TEXT
UOML_VIEWSTREAM

图 15


UOML_DELETE
handle

图 16

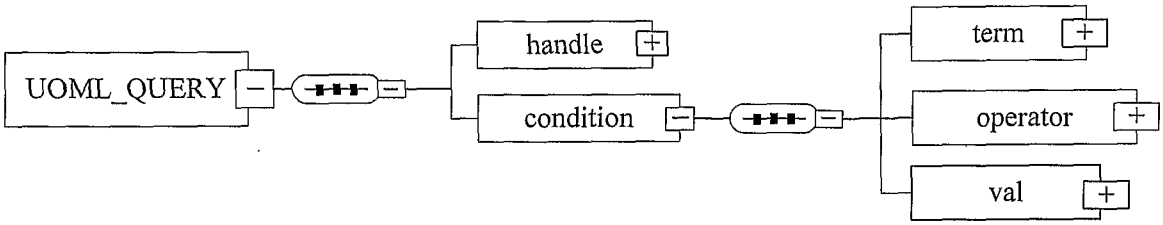

UOML_QUERY
handle
condition
term
operator
val

图 17

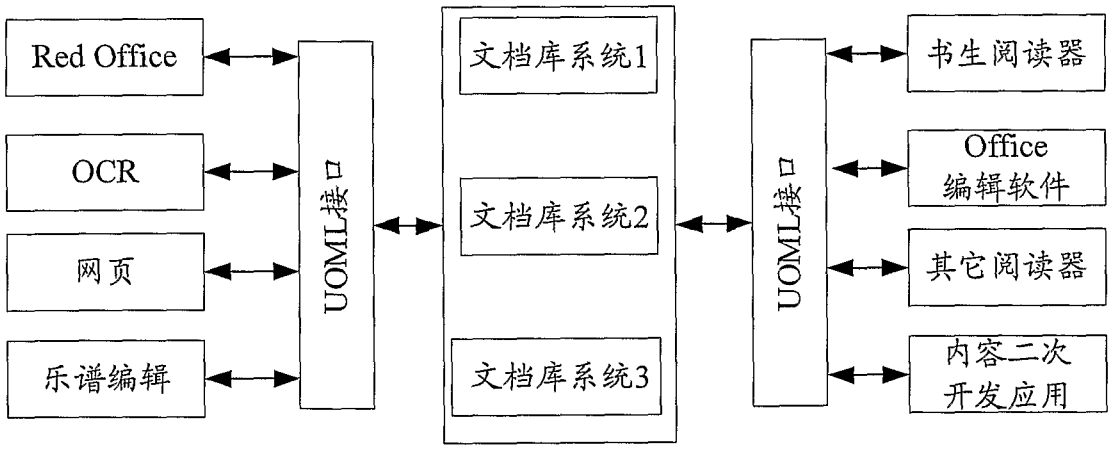

Red Office
OCR
网页
乐谱编辑
UOML接口
文档库系统1
文档库系统2
文档库系统3
UOML接口
书生阅读器
Office
编辑软件
其它阅读器
内容二次
开发应用

图 18

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2006/003297

A. CLASSIFICATION OF SUBJECT MATTER

G06F17/30 (2006.01) i

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)

G06F17/ (2006.01) i

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

CHINESE DOCUMENTS

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI, EPODOC, PAJ, CNPAT, CNKI: DOCUMENT, UNIVERSAL,APPLICATION W LAYER, COMPATIBLE, CREAT+, GENERAT+, CUSTOMIZ+, INSTRUCTION

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US,A1,2003144982(BENEFITNATION)31.Jul.2003(31.07.2003), Description Paragraph[0034-0036],[0046-0047]	1-8, 12, 14-16
X	US,A1,2004205656((BENEFITNATION),14.Oct.2004(14.10.2004), Description Paragraph[0034-0036],[0046-0047]	1-8, 12, 14-16
A	US,A,6006242(BANKERS SYSTEMS, INC.),21.Dec.1999(21.12.1999), the whole document	1-16
A	US,A1,2005050444(PHILIP E.VASEY), 03.Mar.2005(03.03.2005) , the whole document	1-16

☐ Further documents are listed in the continuation of Box C. ☒ 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 application or patent 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
02.Mar.2007 (02.03.2007)	29 · MAR 2007 (29 · 03 · 2007)
Name and mailing address of the ISA/CN The State Intellectual Property Office, the P.R.China 6 Xitucheng Rd., Jimen Bridge, Haidian District, Beijing, China 100088 Facsimile No. 86-10-62019451	Authorized officer WANG, Jingxia Telephone No. 86-10-62085037

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

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2006/003297

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
US,A1,2003144982	31.Jul.2003(31.07.2003)	US,B2,7035837	25.Apr.2006(25.04.2006)
US,A1,2004205656	14.Oct.2004(14.10.2004)	NONE	
US,A,6006242	21.Dec.1999(21.12.1999)	NONE	
US,A1,2005050444	03.Mar.2005（03.03.2005）	GB,A,2405730	09.Mar.2005(09.03.2005)
		WO,A2,2005024656	17.Mar.2005(17.03.2005)

国际检索报告

国际申请号
PCT/CN2006/003297

A. 主题的分类

G06F17/30（2006.01）i

按照国际专利分类表(IPC)或者同时按照国家分类和 IPC 两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

G06F17/（2006.01）i

包含在检索领域中的除最低限度文献以外的检索文献

中文文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用）)

WPI, EPODOC, PAJ, CNPAT, CNKI：文档，通用，创建，产生，定制，应用层，DOCUMENT, UNIVERSAL,APPLICATION W LAYER, COMPATIBLE, CREAT+, GENERAT+, CUSTOMIZ+, INSTRUCTION

C. 相关文件

类　型*	引用文件，必要时，指明相关段落	相关的权利要求
X	US,A1,2003144982（BENEFITNATION），31.7 月 2003（31.07.2003），说明书第[0034-0036]、[0046-0047]段	1-8，12，14-16
X	US,A1,2004205656（BENEFITNATION），14.10 月 2004（14.10.2004），说明书第[0034-0036]、[0046-0047]段	1-8，12，14-16
A	US,A,6006242（BANKERS SYSTEMS, INC.），21.12 月 1999（21.12.1999），全文	1-16
A	US,A1,2005050444（PHILIP E.VASEY），03.3 月 2005（03.03.2005），全文	1-16

☐ 其余文件在 C 栏的续页中列出。　　☒ 见同族专利附件。

* 引用文件的具体类型：

"A" 认为不特别相关的表示了现有技术一般状态的文件

"E" 在国际申请日的当天或之后公布的在先申请或专利

"L" 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件

"O" 涉及口头公开、使用、展览或其他方式公开的文件

"P" 公布日先于国际申请日但迟于所要求的优先权日的文件

"T" 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件

"X" 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性

"Y" 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性

"&" 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
02.3 月 2007（02.03.2007）	29·3月 2007 (29·03·2007)
中华人民共和国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路 6 号 100088 传真号：　(86-10)62019451	受权官员 王京霞 电话号码：(86-10)62085037

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2006/003297

检索报告中引用的专利文件	公布日期	同族专利	公布日期
US,A1,2003144982	31.7 月 2003(31.07.2003)	US,B2,7035837	25.4 月 2006(25.04.2006)
US,A1,2004205656	14.10 月 2004(14.10.2004)	无	
US,A,6006242	21.12 月 1999(21.12.1999)	无	
US,A1,2005050444	03.3 月 2005（03.03.2005）	GB,A,2405730	09.3 月 2005(09.03.2005)
		WO,A2,2005024656	17.3 月 2005(17.03.2005)