[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/21 (2006.01)
G06F 12/00 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200680034204.0

[43] 公开日 2008 年 9 月 17 日

[11] 公开号 CN 101268458A

[22] 申请日 2006.9.21
[21] 申请号 200680034204.0
[30] 优先权
[32] 2005. 9.22 [33] JP [31] 275366/2005
[86] 国际申请 PCT/JP2006/318707 2006.9.21
[87] 国际公布 WO2007/034858 日 2007.3.29
[85] 进入国家阶段日期 2008.3.17
[71] 申请人 佳思腾软件公司
地址 日本德岛县
[72] 发明人 樋浦秀树 本桥大辅

[74] 专利代理机构 北京英赛嘉华知识产权代理有限责任公司
代理人 余 朦 王艳春

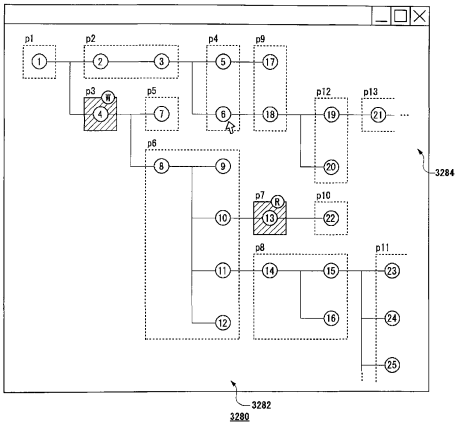
权利要求书 8 页 说明书 53 页 附图 38 页

[54] 发明名称

数据管理装置、数据编辑装置、数据阅览装置、数据管理方法、数据编辑方法以及数据阅览方法

[57] 摘要

由多个用户有效地编辑数据文件。 从结构化文档文件等各种数据文件生成源对象。 从源对象生成多个部分对象。 客户终端下载与用户希望编辑或阅览的数据对应的部分对象。 当获取了编辑权的客户终端编辑本地部分对象时，将表示编辑内容的编辑信息发送到服务器装置。 根据该编辑信息更新源对象。 另外，当其他客户终端下载相同的部分对象时，用于反映编辑内容的编辑信息从服务器装置发送到客户终端。



1. 通过网络与多个客户终端连接的数据管理装置，包括:

源对象生成单元，生成用于操作要被编辑的一组元素数据的源对象，作为包括一组结构化的元素数据和用于访问各元素数据的成员函数的对象;

部分对象生成单元，从所述源对象生成多个部分对象，作为与所述源对象中包括的所述一组元素数据的一部分对应的对象;

访问请求接收单元，从客户终端接收访问请求信息，该访问请求信息用于请求在所述多个部分对象中指定的部分对象的编辑权或者阅览权;

访问权管理单元，根据从多个客户终端接收的访问请求信息，向各客户终端分配每个所述部分对象的编辑权和阅览权;

部分对象发送单元，向请求对所述访问请求信息指定的部分对象进行编辑或阅览的客户终端发送该部分对象的数据，以使得所述客户终端本地保存所述的数据;

编辑信息接收单元，当在获取了所述部分对象的编辑权的客户终端中编辑本地保持的该部分对象时，从获取所述述编辑权的客户终端接收表示所述部分对象的编辑内容的编辑信息;

源对象更新单元，按照接收的所述编辑信息中所示的编辑内容，更新所述源对象的相关部分; 以及

编辑信息发送单元，将表示所述部分对象的编辑内容的编辑信息发送到获取了所述阅览权的客户终端，从而使得所述编辑内容反映在所述客户终端本地保持的部分对象的相关部分中。

2. 根据权利要求1所述的数据管理装置，其中，所述源对象生成单元从由多个标签标识元素数据的结构化文档文件生成所述源对象。

3. 根据权利要求1或2所述的数据管理装置，其中，所述源对象生成单元参照用于将预定数据库中包括的多个元素数据进行结构化的预定

结构定义信息，将所述多个元素数据中将要被编辑的一组元素数据进行结构化，并从所结构化的一组元素数据生成源对象。

4. 根据权利要求1至3任意一项所述的数据管理装置，还包括:

结构化的管理信息保持单元，保持表示源对象中包括的一组元素数据的层次结构的结构化的管理信息;

结构化的管理信息发送单元，将所述结构化的管理信息发送到客户终端；以及

结构化的管理信息更新单元，根据源对象的更新来更新所述结构化的管理信息，

其中，当所述结构化的管理信息所示的一组元素数据的层次结构变化时，所述结构化的管理信息发送单元将更新后的结构管理信息重新发送到所述客户终端。

5. 数据编辑装置，其与管理源对象的外部数据管理装置通过网络连接，该源对象用于操作要被编辑的一组元素数据，其中，所述数据编辑装置包括:

访问请求信息发送单元，将用于请求编辑权的访问请求信息发送到所述数据管理装置，所述访问请求信息根据源对象中包括的一组元素数据的一部分，在所述数据管理装置生成的多个部分对象中指定相应的部分对象;

部分对象接收单元，从所述数据管理装置接收被指定的部分对象的数据;

部分对象保持单元，保持已接收的所述部分对象的数据;

编辑输入单元，接收用户对保持的部分对象的编辑操作;

部分对象更新单元，根据对所述部分对象的编辑操作更新所述部分对象；以及

编辑信息发送单元，将表示所述部分对象的编辑内容的编辑信息发送到所述数据管理装置，从而使得所述编辑内容反映在所述数据管理装置的源对象中。

6. 根据权利要求5所述的数据编辑装置，还包括:

结构化的管理信息接收单元，从所述数据管理装置接收结构化的管理信息，该结构化的管理信息表示源对象中包括的一组元素数据的层次结构;

结构化的管理信息显示单元，显示所述结构化的管理信息;

编辑位置指定单元，接受用户对所显示的结构化的管理信息希望编辑的数据位置的输入;

其中，所述访问请求信息发送单元参照所述结构化的管理信息识别与所指定的位置对应的部分对象，并发送用于请求所识别的部分对象的编辑权的访问请求信息。

7. 根据权利要求6所述的数据编辑装置，其中，所述源对象由多个数据管理装置分割为多个来保持，并且所述结构化的管理信息还表示所述源对象中包括的各元素数据的实体位于哪个数据管理装置中，

所述访问请求信息发送单元参照所述结构化的管理信息识别保持被指定为要被编辑的部分对象的数据管理装置，并向所识别出的数据管理装置发送访问请求信息。

8. 根据权利要求5至7任意一项所述的数据编辑装置，其中，所述部分对象保持单元能够保持多个部分对象，并包括:

第1保持单元，保持要被编辑的部分对象;

第2保持单元，当部分对象未被编辑时保存在所述第2保持单元中;

当指定为要被编辑的部分对象保持在所述第2保持单元中时，所述访问请求信息发送单元将该部分对象从所述第2保持单元加载到所述第1保持单元中，而不发送访问请求信息。

9. 数据阅览装置，与管理源对象的外部数据管理装置通过网络连接，该源对象用于操作要被编辑的一组元素数据，所述数据阅览装置包括:

访问请求信息发送单元，将用于请求阅览权的访问请求信息发送到所述数据管理装置，所述访问请求信息根据源对象中包括的一组元素数据的一部分，在所述数据管理装置生成的多个部分对象中指定相应的部分对象；

部分对象接收单元，从所述数据管理装置接收所指定的部分对象的数据；

部分对象保持单元，保持已接收的部分对象的数据；

编辑信息接收单元，当与所保持的部分对象相当的、源对象的一部分在所述数据管理装置中更新时，接收从所述数据管理装置发送的编辑信息以表示所更新的内容；以及

部分对象更新单元，根据所述编辑信息更新所述保持的部分对象。

10. 根据权利要求9所述的数据阅览装置，还包括：

结构化的管理信息接收单元，从所述数据管理装置接收结构化的管理信息，该结构化的管理信息表示源对象中包括的一组元素数据的层次结构；

结构化的管理信息显示单元，显示所述结构化的管理信息；以及

阅览位置指定单元，从用户接收输入以指定所显示的结构化的管理信息的所述用户期望阅览的数据位置；

其中所述访问请求信息发送单元参照所述结构化的管理信息识别与所指定数据位置对应的部分对象，以及发送用于请求所识别的部分对象的阅览权的访问请求信息。

11. 根据权利要求10所述的数据阅览装置，其中，源对象由多个数据管理装置分割为多个来保持，并且所述结构化的管理信息还表示所述源对象中包括的各元素数据的实体位于哪个数据管理装置中，

所述访问请求信息发送单元参照所述结构化的管理信息识别保持有被指定为要被阅览的部分对象的数据管理装置，并向所识别的数据管理装置发送访问请求信息。

12. 根据权利要求9至11任意一项所述的数据阅览装置，其中，所述部分对象保持单元能够保持多个部分对象，其进一步包括:

第1保持单元，保持要被阅览的部分对象;

第2保持单元，当部分对象未被阅览时保存在所述第2保持单元中;

当指定为要被阅览的部分对象保持在所述第2保持单元中时，所述访问请求信息发送单元将该部分对象从所述第2保持单元加载到所述第1保持单元中，而不发送访问请求信息。

13. 数据管理方法，所述方法在通过网络与多个客户终端连接的装置中执行，并包括:

生成用于操作要被编辑的一组元素数据的源对象，作为包括一组结构化的元素数据和用于访问各元素数据的成员函数的对象;

从所述源对象生成多个部分对象，作为与所述源对象中包括的所述一组元素数据的一部分对应的对象;

从客户终端接收访问请求信息，该访问请求信息用于请求在所述多个部分对象中指定的部分对象的编辑权或者阅览权;

根据从多个客户终端接收的访问请求信息，向各客户终端分配每个所述部分对象的编辑权和阅览权;

向请求对所述访问请求信息指定的部分对象进行编辑或阅览的客户终端发送所述访问请求信息指定的部分对象的数据，以使得所述客户终端本地保存所述的数据;

当在获取了所述部分对象的编辑权的客户终端中编辑本地保持的该部分对象时，从获取所述述编辑权的客户终端接收表示所述部分对象的编辑内容的编辑信息;

按照接收的所述编辑信息中所示的编辑内容，更新所述源对象的相关部分; 以及

将表示所述部分对象的编辑内容的编辑信息发送到获取了所述阅览权的客户终端，从而使得所述编辑内容反映在所述客户终端本地保持的部分对象的相关部分中。

14. 数据编辑方法，在与管理源对象的外部数据管理装置通过网络连接的装置中执行，该源对象用于操作要被编辑的一组元素数据，所述方法包括:

将用于请求编辑权的访问请求信息发送到所述数据管理装置，所述访问请求信息根据源对象中包括的一组元素数据的一部分，在所述数据管理装置生成的多个部分对象中指定相应的部分对象;

从所述数据管理装置接收被指定的部分对象的数据;

保持已接收的所述部分对象的数据;

接收用户对保持的部分对象的编辑操作;

根据对所述部分对象的编辑操作更新所述部分对象；以及

将表示所述部分对象的编辑内容的编辑信息发送到所述数据管理装置，从而使得所述编辑内容反映在所述数据管理装置的源对象中。

15. 数据阅览方法，在与管理源对象的外部数据管理装置通过网络连接的装置中执行，该源对象用于操作要被编辑的一组元素数据，所述方法包括:

将用于请求阅览权的访问请求信息发送到所述数据管理装置，所述访问请求信息根据源对象中包括的一组元素数据的一部分，在所述数据管理装置生成的多个部分对象中指定相应的部分对象;

从所述数据管理装置接收所指定的部分对象的数据;

保持已接收的部分对象的数据;

当与所保持的部分对象相当的、源对象的一部分在所述数据管理装置中更新时，接收从所述数据管理装置发送的编辑信息以表示所更新的内容；以及

根据所述编辑信息更新所述保持的部分对象。

16. 数据管理程序，所述数据管理程序为在通过网络与多个客户终端连接的装置中执行的计算机程序并包括:

源对象生成模块，生成用于操作要被编辑的一组元素数据的源对象，作为包括一组结构化的元素数据和用于访问各元素数据的成员函数

的对象;

部分对象生成模块，从所述源对象生成多个部分对象，作为与所述源对象中包括的所述一组元素数据的一部分对应的对象;

访问请求接收模块，从客户终端接收访问请求信息，该访问请求信息用于请求在所述多个部分对象中指定的部分对象的编辑权或者阅览权;

访问权管理模块，根据从多个客户终端接收的访问请求信息，向各客户终端分配每个所述部分对象的编辑权和阅览权;

部分对象发送模块，向请求对所述访问请求信息指定的部分对象进行编辑或阅览的客户终端发送该部分对象的数据，以使得所述客户终端本地保存所述的数据;

编辑信息接收模块，当在获取了所述部分对象的编辑权的客户终端中编辑本地保持的该部分对象时，从获取所述述编辑权的客户终端接收表示所述部分对象的编辑内容的编辑信息;

源对象更新模块，按照接收的所述编辑信息中所示的编辑内容，更新所述源对象的相关部分；以及

编辑信息发送模块，将表示所述部分对象的编辑内容的编辑信息发送到获取了所述阅览权的客户终端，从而使得所述编辑内容反映在所述客户终端本地保持的部分对象的相关部分中。

17. 数据编辑程序，所述是在与管理源对象的外部数据管理装置通过网络连接的装置中执行的计算机程序，该源对象用于操作要被编辑的一组元素数据，所述计算机程序包括:

将用于请求编辑权的访问请求信息发送到所述数据管理装置的模块，所述访问请求信息根据源对象中包括的一组元素数据的一部分，在所述数据管理装置生成的多个部分对象中指定相应的部分对象;

从所述数据管理装置接收被指定的部分对象的数据的模块;

保持已接收的所述部分对象的数据的模块;

接收用户对保持的部分对象的编辑操作的模块;

根据对所述部分对象的编辑操作更新所述部分对象的模块；以及

将表示所述部分对象的编辑内容的编辑信息发送到所述数据管理装置，从而使得所述编辑内容反映在所述数据管理装置的源对象中的模块。

18. 数据阅览程序，所述数据阅览程序是在与管理源对象的外部数据管理装置通过网络连接的装置中执行的计算机程序，该源对象用于操作要被编辑的一组元素数据，所述数据阅览程序包括:

将用于请求阅览权的访问请求信息发送到所述数据管理装置的模块，所述访问请求信息根据源对象中包括的一组元素数据的一部分，在所述数据管理装置生成的多个部分对象中指定相应的部分对象;

从所述数据管理装置接收所指定的部分对象的数据的模块;

保持已接收的部分对象的数据的模块;

当与所保持的部分对象相当的、源对象的一部分在所述数据管理装置中更新时，接收从所述数据管理装置发送的编辑信息以表示所更新的内容的模块；以及

根据所述编辑信息更新所述保持的部分对象的模块。

数据管理装置、数据编辑装置、数据阅览装置、数据管理方法、数据编辑方法以及数据阅览方法

技术领域

本发明涉及用于编辑数据的技术，特别涉及用于由多个用户同步编辑数据的技术。

背景技术

近年来，伴随着计算机的普及和网络技术的进步，通过网络的电子信息交换变得频繁。由此，以前基于纸完成的大多数事务处理逐渐替换为基于网络的处理。

在企业中，将个人知识和信息在整个组织中使用的、所谓的知识管理也变成重要的经营手段。在多数企业中，公司具有内部数据库系统，将来自员工的信息在转换为电子文件后蓄积。另一方面，员工也通过网络访问蓄积在该公司内部数据库中的文件。由此，在整体上提高组织的业务效率。

在近年中，蓄积在这种数据库中的文件使用被称作XML（eXtensible Markup Language: 扩展标记语言）的语言制作的例子也逐渐变多。XML是可定义文档文件中包括的数据层次结构的语言。由XML制作的文档文件，通过另外提供的显示布局文件以网页形式显示。即XML能够将数据结构及其显示布局作为不同的内容处理。XML文档文件是根据由文档型定义等定义的词汇（标签集）制作。

XML作为适合通过网路等与他人共享数据的形式而备受关注。近年来，开发有用于制作、显示、编辑XML文档文件的各种应用软件（例如参照专利文献1）。

专利文献1：特开2001-290804号公报

发明内容

发明要解决的课题

由标签集定义数据结构的结构化文档文件，如XML文档文件，与数据库的亲和性高。另外，当将结构化文档中包括的数据对象化时，可进行更有效的数据访问。对象不仅包括在将数据进行结构化后作为成员变量的数据，而且还包括作为访问成员变量的接口的成员函数。访问成员变量的方法是由成员函数严格规定，因此容易保护数据，并且能够进行有效的数据访问。对象也容易适应计算机程序进行的处理。

但是，与源数据的数据量相比，对象的数据量变得相当大。特别是，当从大的源数据生成的对象部署在存储器中时，存储器的存储容量被压缩，反过来处理效率变得恶化。

本发明是鉴于这种状况而做出的，其主要目的在于提供一种用于多个用户有效地操作从源数据生成的对象的技术。

解决课题的方案

本发明一个实施方式的数据管理装置是通过网络与多个客户终端连接的装置。该装置包括：源对象生成单元，生成用于操作要被编辑的一组元素数据的源对象，作为包括一组结构化的元素数据和用于访问各元素数据的成员函数的对象；部分对象生成单元，从所述源对象生成多个部分对象，作为与所述源对象中包括的所述一组元素数据的一部分对应的对象；访问请求接收单元，从客户终端接收访问请求信息，该访问请求信息用于请求在所述多个部分对象中指定的部分对象的编辑权或者阅览权；访问权管理单元，根据从多个客户终端接收的访问请求信息，向各客户终端分配每个所述部分对象的编辑权和阅览权；部分对象发送单元，向请求对所述访问请求信息指定的部分对象进行编辑或阅览的客户终端发送该部分对象的数据，以使得所述客户终端本地保存所述的数据；编辑信息接收单元，当在获取了所述部分对象的编辑权的客户终端中编辑本地保持的该部分对象时，从获取所述述编辑权的客户终端接收表示所述部分对象的编辑内容的编辑信息；源对象更新单元，按照接收的所述编辑信息中所示的编辑内容，更新所述源对象的相关部分；以及编辑信息发送单元，将表示所述部分对象的编辑内容的编辑信息发送到

获取了所述阅览权的客户终端，从而使得所述编辑内容反映在所述客户终端本地保持的部分对象的相关部分中。

部分对象还可以是包括在源对象中包括的元素数据的一部分，和用于访问这些元素数据的成员函数的对象。编辑信息例如可以是显示通过成员函数访问部分对象的访问操作以及访问内容的信息。或者，编辑信息也可以是由编辑操作更新后的部分对象数据本身。编辑信息是至少用于将部分对象的更新反映在源对象所必须的信息。对于发送到客户终端的编辑信息也是如此。

源对象生成单元还可以从由多个标签确定的元素数据的结构化文档文件生成源对象。

源对象生成单元参照用于将预定数据库中包括的多个元素数据进行结构化的预定结构定义信息，将多个元素数据中要被编辑的一组元素数据进行结构化，并从结构化的一组元素数据生成源对象。

该装置也可以进一步具备：结构化的管理信息保持单元，保持表示源对象中包括的一组元素数据的层次结构的结构管理信息；结构化的管理信息发送单元，将结构化的管理信息发送到客户终端；结构化的管理信息更新单元，根据源对象的更新来更新结构化的管理信息。

当结构化的管理信息所示的一组元素数据的层次结构变化时，结构化的管理信息接收单元可以将更新后的结构管理信息重新发送到客户终端。

本发明另一实施方式的数据编辑装置与管理源对象的外部数据管理装置通过网络连接。所述的数据编辑装置包括：访问请求信息发送单元，将用于请求编辑权的访问请求信息发送到所述数据管理装置，所述访问请求信息根据源对象中包括的一组元素数据的一部分，在所述数据管理装置生成的多个部分对象中指定相应的部分对象；部分对象接收单元，从所述数据管理装置接收被指定的部分对象的数据；部分对象保持单元，保持已接收的所述部分对象的数据；编辑输入单元，接收用户对保持的部分对象的编辑操作；部分对象更新单元，根据对所述部分对象的编辑操作更新所述部分对象；以及编辑信息发送单元，将表示所述部分对象的编辑内容的编辑信息发送到所述数据管理装置，从而使得所述

编辑内容反映在所述数据管理装置的源对象中

该装置也可以进一步包括：结构化的管理信息接收单元，从所述数据管理装置接收结构化的管理信息，该结构化的管理信息表示源对象中包括的一组元素数据的层次结构；结构化的管理信息显示单元，显示所述结构化的管理信息；编辑位置指定单元，接受用户对所显示的结构化的管理信息希望编辑的数据位置的输入。所述访问请求信息发送单元参照所述结构化的管理信息识别与所指定的位置对应的部分对象，并发送用于请求所识别的部分对象的编辑权的访问请求信息。访问请求信息发送单元也可以参照结构管理信息确定与指定位置对应的部分对象，发送用于请求已确定的部分对象的编辑权的访问请求信息。

源对象也可以由多个数据管理装置分割为多个来保持，并且结构化的管理信息还表示源对象中包括的各元素数据的实体位于哪个数据管理装置中。

而且，该装置的所述访问请求信息发送单元参照所述结构化的管理信息识别保持被指定为要被编辑的部分对象的数据管理装置，并向所识别出的数据管理装置发送访问请求信息。

该装置的部分对象保持单元可保持多个部分对象。部分对象保持单元也可以包括：第1保持单元，保持要被阅览的部分对象；以及第2保持单元，当部分对象未被阅览时保存在所述第2保持单元中。当指定为要被阅览的部分对象保持在所述第2保持单元中时，所述访问请求信息发送单元将该部分对象从所述第2保持单元加载到所述第1保持单元中，而不发送访问请求信息。

本发明的另一实施方式的数据阅览装置也与管理源对象的外部数据管理装置通过网络连接。该装置包括：访问请求信息发送单元，将用于请求阅览权的访问请求信息发送到所述数据管理装置，所述访问请求信息根据源对象中包括的一组元素数据的一部分，在所述数据管理装置生成的多个部分对象中指定相应的部分对象；部分对象接收单元，从所述数据管理装置接收所指定的部分对象的数据；部分对象保持单元，保持已接收的部分对象的数据；编辑信息接收单元，当与所保持的部分对象相当的、源对象的一部分在所述数据管理装置中更新时，接收从所述数

据管理装置发送的编辑信息以表示所更新的内容；以及部分对象更新单元，根据所述编辑信息更新所述保持的部分对象。

该装置也可以进一步包括：结构化的管理信息接收单元，从所述数据管理装置接收结构化的管理信息，该结构化的管理信息表示源对象中包括的一组元素数据的层次结构；结构化的管理信息显示单元，显示所述结构化的管理信息；以及阅览位置指定单元，从用户接收输入以指定所显示的结构化的管理信息的所述用户期望阅览的数据位置。所述访问请求信息发送单元参照所述结构化的管理信息识别与所指定数据位置对应的部分对象，以及发送用于请求所识别的部分对象的阅览权的访问请求信息。

源对象也可以由多个数据管理装置分割为多个来保持，并且结构化的管理信息还表示源对象中包括的各元素数据的实体位于哪个数据管理装置中。该装置的所述访问请求信息发送单元参照所述结构化的管理信息识别保持有被指定为要被阅览的部分对象的数据管理装置，并向所识别的数据管理装置发送访问请求信息。

应该认识到，以上结构元素的任意组合，以及本发明的表现在方法、装置、系统、记录介质、计算机程序等之间变换的内容作为本发明方式也是有效的。

发明的效果

根据本发明，多个用户容易有效地操作从源数据生成的对象。

附图说明

图1是与前提技术相关的文档处理装置的构成示意图；

图2示出了成为处理对象的XML文档的实施例；

图3示例性地示出了将图2所示的XML文档映射为由HTML描述的表；

图4(a)示出了用于将图2所示的XML文档映射为图3所示的表的定义文件的实施例；

图4(b)示出了用于将图2所示的XML文档映射为图3所示的表的定义

文件的实施例;

图5示出了通过图3所示的对应关系将图2所示的成绩管理词描述的XML文档映射为HTML并显示的屏幕的实施例;

图6示出了为使用户创建定义文件而定义文件生成单元提供给用户的图形用户界面的实施例;

图7示出了利用定义文件生成单元创建的屏幕布局的另一个实施例;

图8示出了文档处理装置提供的XML文档的编辑屏幕的实施例;

图9示出了利用文档处理装置编辑的XML文档的另一实施例;

图10示出了显示图9所示文档的屏幕的实施例;

图11(a)示出了文档处理系统的基本构成;

图11(b)示出了文档处理系统的总体框图;

图11(c)示出了文档处理系统的总体框图;

图12示出了文档管理单元的细节;

图13示出了词汇连接子系统的细节;

图14示出了程序调用单元与其它组件之间的关系的细节;

图15示出了由程序调用单元载入的应用程序服务的结构细节;

图16示出了核心组件的细节;

图17示出了文档管理单元的细节;

图18示出了撤消框架和撤消命令的细节;

图19示出了在文档处理系统中载入文档的操作;

图20示出了文档及其表现的实施例;

图21示出了模型和控制器的关系;

图22示出了插件子系统、词汇连接与连接器的细节;

图23示出了VCD文件的例子;

图24示出了用于将复合文档载入到文档处理系统中的操作;

图25示出了用于将复合文档载入到文档处理系统中的操作;

图26示出了用于将复合文档载入到文档处理系统中的操作;

图27示出了用于将复合文档载入到文档处理系统中的操作;

图28示出了用于将复合文档载入到文档处理系统中的操作;

图29示出了命令流;

图30是用于说明本实施例中的数据处理内容的示意图;

图31是用于说明对某部分对象提出多个访问要求时的控制方法的序列图;

图32是数据管理装置的功能框图;

图33是数据处理终端的功能框图;

图34是用于说明从源对象生成部分对象的过程的示意图;

图35是结构化的管理文件的数据结构图;

图36是在数据处理终端中选择要访问的数据时的屏幕图。

符号说明

20 文档处理装置	22 主控单元	24 编辑单元
30 DOM单元	32 DOM提供单元	34 DOM生成单元
36 输出单元	40 CSS单元	42 CSS分析单元
44 CSS提供单元	46 呈现单元	50 HTML单元
52、62 控制单元	54、64 编辑单元	56、66 显示单元
60 SVG单元	80 VC单元	82 映射单元
84 定义文件获取单元		86 定义文件生成单元
3100文档管理装置		3110 用户接口处理单元
3120 通信单元		3130 数据处理单元
3140 数据保存单元		3142 访问请求接收单元
3144 部分对象发送单元		3146 结构化的管理文件发送单元
3148 编辑信息接收单元		3150 编辑信息发送单元
3152 对象处理单元		3154 访问权管理单元
3156 源对象生成单元		3158 源对象更新单元
3160 部分对象生成单元		3162 结构化的管理文件处理单元
3166 源数据保持单元		3168 对象保持单元
3170 访问权信息保持单元		3172 结构化的管理文件保持单元
3200 数据处理终端		3210 用户接口处理单元
3220 通信单元		3230 数据处理单元

3240 数据保存单元 3242 访问请求发送单元
3244 部分对象接收单元 3246 结构化的管理文件接收单元
3248 编辑信息发送单元 3250 编辑信息接收单元
3252 部分对象更新单元 3254 访问权管理单元
3256 部分对象保持单元 3258 第1保持单元
3260 第2保持单元 3262 结构化的管理文件保持单元
3264 访问标记保持单元

具体实施方式

（前提技术）

图1示出了与前提技术相关的文档处理装置20的结构。文档处理装置20对结构化的文档进行处理，该文档中的数据被分为具有分级结构的多个构成元素。在本前提技术中以对作为一种结构化文档的XML文档进行处理为例来说明。文档处理装置20包括主控单元22、编辑单元24、DOM（文档对象模块）单元30、CSS（层叠样式表）单元40、HTML（超文本标记语言）单元50、SVG（可缩放矢量图形）单元60以及作为变换单元一个示例的VC（词汇连接）单元80。在硬件组件方面，这些单元结构可由任意计算机的CPU、存储器、载入存储器中的程序等来实现。这里，描述了由它们的协作而实现的功能模块。因此，本领域技术人员能够理解，这些功能模块可仅通过硬件的方式、仅通过软件的方式或通过二者相结合的方式等以多种形式来实现。

主控单元 22 提供插件的载入或提供执行命令的框架。编辑单元 24 提供了用于编辑 XML 文档的框架。文档处理装置 20 中的文档的显示和编辑功能是通过插件来实现的，而必要的插件是根据所处理的文档类型、通过主控单元 22 或编辑单元 24 来载入的。主控单元 22 或编辑单元 24 通过参考作为处理对象的 XML 文档的命名空间，来确定哪个或哪些词汇描述了待处理的 XML 文档的内容，并且载入与所确定的词汇对应的用于显示和编辑的插件，从而执行显示和编辑。例如，在文档处理装置 20 中，对 HTML 文档进行显示和编辑的 HTML 单元 50、以及对 SVG 文档进行显示和编辑的 SVG 单元 60 等被实现为用于各词汇（标签

集）的显示系统和编辑系统的插件，以分别在对 HTML 文档进行编辑时载入 HTML 单元 50，和在对 SVG 文档进行编辑时载入 SVG 单元 60。如以下将描述的那样，在要对既包括 HTML 又包括 SVG 组件的复合文档进行处理时，既载入 HTML 单元 50 又载入 SVG 单元 60。

通过实现以上结构，由于用户能够仅选择并安装必要的功能，然后可以增加或删除适当的功能，因此，能够有效利用记录媒介的存储区域（例如硬盘），并且在执行程序的时候还能够避免存储器的浪费。此外，由于这一结构有利于性能扩展，因此开发者自己能够以插件的形式处理新的词汇，因而能够促进开发过程；用户也能够通过增加插件而以较低成本容易地增加功能。

编辑单元24通过用户接口从用户处接收编辑指令的事件。编辑单元24将该事件通知适当的插件等，并对包括事件的重做（redo）以及撤消（undo）等的处理进行控制。

DOM 单元 30 包括 DOM 提供单元 32、DOM 生成单元 34 以及输出单元 36。DOM 单元 30 实现了与文档对象模型（DOM）相符的功能，所述文档对象模型被定义以提供用于处理 XML 文档形式的数据的访问方法。DOM 提供单元 32 是满足由编辑单元 24 定义的接口的 DOM 的实现。DOM 生成单元 34 从 XML 文档创建 DOM 树。如以下将描述的那样，当通过 VC 单元 80 将待处理的 XML 文档映射为其它词汇时，创建与映射源中的 XML 文档相对应的源树，以及与映射目的中的 XML 文档相对应的目的树。例如，在编辑结束时，输出单元 36 输出作为 XML 文档的 DOM 树。

CSS单元40提供与CSS相符的显示功能，并包括CSS分析单元42、CSS提供单元44以及呈现单元46。CSS分析单元42具有用于分析CSS语法的分析功能。CSS提供单元44是CSS对象的实现，并对DOM树执行CSS的层叠处理。呈现单元46是CSS的呈现引擎，并用来显示利用CSS设置的以诸如HTML等的词汇描述的文档。

HTML 单元 50 对以 HTML 描述的文档进行显示或编辑。SVG 单元 60 对以 SVG 描述的文档进行显示或编辑。这些显示/编辑系统以插件的形式实现，各个系统包括对文档进行显示的显示单元（在本文中还称为

“画布（Canvas）”56、66，发送和接收包括编辑命令的事件的控制单元 （在本文中还称为“Editlet”）52、62，以及在接收到编辑命令时对DOM 进行编辑的编辑单元（在本文中还称为“区（zone）”54、64。在控制单元 52 或 62 从外部源接收到用于 DOM 树的编辑命令时，编辑单元 54 或 64 修改 DOM 树，而显示单元 56 或 66 更新显示。这些单元具有与被称作 MVC（Model-View-Controllers，模型－视图－控制器）的框架相类似的结构，通常，显示单元 56 及 66 对应于“视图（View）”，控制单元 52 及 62 对应于“控制器（Controller）”，而编辑单元 54 及 64 和 DOM 实体对应于“模型（Model）”。在本前提技术的文档处理装置 20 中，不仅能够以树型视图显示格式来编辑 XML 文档，而且能够根据相应的词汇来完成编辑。例如，HTML 单元 50 提供能够用来以一种类似于 Word 处理器的方法对 HTML 文档进行编辑的用户界面，而 SVG 单元 60 也提供了能够用于以一种类似于图像绘制工具的方法对 SVG 文档进行编辑的用户界面。

VC 单元 80 包括映射单元 82、定义文件获取单元 84 以及定义文件生成单元 86。VC 单元 80 能够将以某词汇描述的文档映射为另一给定词汇，从而提供了一种能够通过与被映射的词汇相对应的显示或编辑插件来显示或编辑文档的框架。在本前提技术中，该功能被称为 VC（Vocabulary Connection，词汇连接）。定义文件获取单元 84 获取描述了映射定义的脚本文件。该定义文件逐个节点地描述了节点间的对应（连接）。此时，可规定各节点的元素值或属性值是否可以编辑。也可描述使用了节点的元素值或属性值的运算表达式。这些功能将在稍后进行描述。映射单元 82 使得 DOM 生成单元 34 通过参考 VC 定义文件获取单元 84 已经获取的脚本文件来创建目的树，以管理源树与目的树之间的对应关系。定义文件生成单元 86 为用户提供图形用户界面，以创建定义文件。

VC单元80对源树与目的树之间的连接进行监控。当VC单元80通过由负责显示的插件提供的用户接口从用户处接收编辑指令时，它首先修改源树的相关节点。因此，DOM单元30将发出指示源树已经被修改的变化事件。然后，VC单元80接收该变化事件，并修改目的树中对应于被修

改的节点的节点，以使得目的树与源树的修改同步。当显示/编辑目的树的插件（例如HTML单元50）接收了指示目的树已经被修改的变化事件时，该插件通过参考被修改的目的树而对显示进行更新。通过执行将词汇转换为另一主要词汇的上述结构，即使是以少数用户使用的局部词汇来描述文档，也能够显示该文档，并为其提供编辑环境。

以下对文档处理装置 20 显示或编辑文档的操作进行描述。当文档处理装置 20 载入待处理的文档时，DOM 生成单元 34 从 XML 文档创建 DOM 树。主控单元 22 或编辑单元 24 通过参考待处理的 XML 文档的命名空间来对描述 XML 文档的词汇进行判别。如果与词汇相对应的插件安装在文档处理装置 20 中，则该插件被载入以显示/编辑文档。另一方面，如果插件并未安装在其中，则进行检查以确认是否存在映射的定义文件。如果存在定义文件，则定义文件获取单元 84 获取该定义文件，并根据定义创建目的树，以使得能够通过与要被映射的词汇相对应的插件来显示/编辑文档。如果该文档是包含多个词汇的复合文档，如后面所述，则通过与各词汇相对应的插件来显示/编辑该文档的相关部分。如果不存在定义文件，则显示文档的源或树型结构，并在显示屏中进行编辑。

图2示出了作为处理对象的XML文档的例子。该XML文档用于管理学生的成绩数据。作为XML文档的上部节点的构成元素“成績”包括：在“成績”下方为各个学生设置的多个元素“生徒”。构成元素“生徒”具有属性值“名前”，以及子元素 “国語”（日语）、“数学”、“理科”以及“社会”（社会科学）。属性值“名前”存储学生的姓名。构成元素“国語”、“数学”、“理科”和“社会”分别存储日语、数学、自然科学和社会科学的成绩。例如，姓名为“A”的学生的成绩是：日语为“90”、数学为“50”、自然科学为“75”以及社会科学为“60”。下文中，该文档中使用的词汇（标签集）被称作“成绩管理词汇”。

由于本前提技术的文档处理装置20不具有与成绩管理词汇的显示和/或编辑相对应的插件，因此，将使用以上描述的VC功能，而不使用源显示和树显示的其它显示方法来显示该文档。也就是说，通过准备定义文件，使得成绩管理词汇可映射为已具有插件的另一词汇，例如HTML或

SVG。下面将要进行的说明是在假设已经具备了定义文件的情况下进行的，不过对于用户本身用以创建定义文件所必需的用户界面将在后面描述。

图3示出了将图2中所示的XML文档映射为以HTML描述的表的例子。在图3所示的例子中，使以成绩管理词汇描述的“生徒”节点与以HTML描述的表（“TABLE”节点）的行（“TR”节点）相对应。各行的第一列与属性值“名前”相对应，第二列与“国語”节点的元素值相对应，第三列与“数学”节点的元素值相对应，第四列与“理科”节点的元素值相对应，而第五列与“社会”节点的元素值相对应。因此，图2所示的XML文档能以HTML的列表格式来显示。此外，这些属性值和元素值被指定为能够编辑，以使得用户能够使用HTML单元50的编辑功能在利用HTML显示的屏幕上对这些值进行编辑。在第六列中，指定了用来计算日语、数学、自然科学以及社会科学的分数的加权平均的运算表达式，并显示每个学生的分数的平均值。以这种方式，通过在定义文件中指定运算表达式来完成更灵活的显示，从而提高用户在进行编辑时的便利性。另外，将对第六列的编辑指定为不允许，以使得不能单独对平均值本身进行编辑。因此，在映射定义中，能够指定可编辑或不能编辑，以避免用户可能的错误操作。

图4(a)和4(b)表示定义文件的例子，以将图2所示的XML文档映射为图3所示的表。该定义文件通过被定义用于和定义文件一起使用的脚本语言来描述。在图4(a)和4(b)所示的例子中，“生徒の追加”和“生徒の削除”被定义为命令，并分别涉及将节点“生徒”插入源树中的操作以及将节点“生徒”从源树中删除的操作。该定义文件以模板的形式描述了诸如“名前”和“国語”的标题显示于表的第一行中，而节点“生徒”的内容显示于第二行及其随后的行中。在显示节点“生徒”内容的模板中，包含“text-of”的项表示允许进行编辑，而包含“value-of”的项表示不允许进行编辑。在这些显示了节点“生徒”内容的行中，在第六列中描述了运算表达式“(src: 国語 + src: 数学 + src: 理科 + src:社会) div 4”。这意味着显示学生成绩的平均值。

图5示出了将图2所示的由成绩管理词汇描述的XML文档利用图3所

示的对应关系映射为HTML以使其显示在显示屏上时，显示屏的一个例子。在表90各行中从左至右显示的是各学生的姓名、日语成绩、数学成绩、自然科学成绩、社会科学成绩及其平均值。用户能够在该屏幕上对XML文档进行编辑。例如，当第二行第三列中的值变为“70”时，源树中与该节点相对应的元素值（亦即学生“B”的数学成绩）变为“70”。此时，为了使目的树与源树一致，目的树的相应部分因此而改变，从而使得HTML单元50能够根据改变的目的树来对显示进行更新。因此，学生“B”的数学成绩变为“70”，而平均值相应地变为“55”。

在图5所示的屏幕上，例如“生徒の追加”和“生徒の削除”的命令被显示为菜单，如图4(a)、(b)所示的定义文件中所定义的那样。当用户从这些命令中选择一个命令时，节点“生徒”增加至源树中或从源树中删除。以这种方式，利用根据本前提技术的文档处理装置20，不仅能够对分级结构末端中的组件的元素值进行编辑，而且能够对该分级结构进行编辑。具有上述树型结构的编辑功能能够以命令的形式显现给用户。此外，增加或删除表中的行的命令可例如与增加或删除节点“生徒”的操作相关。嵌入其它词汇中的命令可显现给用户。该表可用作输入模板，以使得对于新学生的成绩数据能够以填空的方式来增加。如上所述，在使用HTML单元50的显示/编辑功能的同时，以成绩管理词汇描述的文档可通过VC功能来编辑。

图6示出了由定义文件生成单元86显现给用户的图形用户界面的例子，以使用户能够创建定义文件。待映射的XML文档在屏幕的左侧区域91显示为树。被映射成的XML文档的屏幕布局显示在屏幕的右侧区域92中。该屏幕布局可通过HTML单元50来编辑，用户在屏幕的右侧区域92中确定并创建用于对文档进行显示的屏幕布局。然后，例如，使用诸如鼠标等的指示设备将屏幕的左侧区域91中显示的XML文档的待映射的节点拖动并放置到屏幕的左侧区域91中的HTML屏幕布局中，以指定映射源处的节点与映射目的处的节点之间的连接。例如，当作为元素“生徒”的子元素的“数学”被放置到HTML屏幕上的表90中第一行与第三列的交叉处时，“数学”节点与第三列中的 “TD”节点之间建立连接。各节点均如此被指定为可编辑或者不可编辑。此外，可在显示屏中嵌入运算表达

式。当完成屏幕编辑时，定义文件生成单元86创建描述屏幕布局与节点之间的连接的定义文件。

已经开发出了能够处理主要词汇（例如XHTML（可扩展超文本标记语言）、MathML（数学标记语言）以及SVG（可缩放矢量图形））的浏览器或编辑器。但是，不可能开发出适于以自创词汇描述的所有文档（例如图2中所示的文档）的浏览器或编辑器。然而，如果如上所述创建了用于映射为其它词汇的定义文件，那么以自创词汇描述的文档就能够使用VC功能来显示和/或编辑，而无需不断开发新的浏览器或编辑器。

图7示出了由定义文件生成单元86创建的屏幕布局的另一例子。在图7所示的例子中，在屏幕上产生表90和圆形图93用于显示以成绩管理词汇描述的XML文档。圆形图93以SVG描述。如以下将讨论的那样，由于根据本前提技术的文档处理装置20能够对在单个XML文档内以多个词汇描述的复合文档进行处理，因此，如该例子所示，以HTML描述的表90以及以SVG描述的圆形图93能够显示在同一屏幕上。

图8示出了用于由文档处理装置20处理的XML文档的编辑屏幕的一例。在图8所示的例子中，单个屏幕被分割为多个区域，而待处理的XML文档在各个区域以多种不同显示格式显示。该文档的源在区域94中显示，该文档的树结构在区域95中显示，而以图5所示的HTML描述的表在区域96中显示。该文档在这些区域中的任意区域均可被编辑，当用户对这些区域中的任意区域的内容进行编辑时，源树将被相应修改，从而负责各屏幕显示的插件将更新屏幕，以反映源树的变更。具体而言，负责显示对应编辑屏幕的插件的显示单元被预先注册为变化事件的监听器，所述变化事件提供源树中发生了改变的通知。当源树被任意插件或VC单元80修改时，显示编辑屏幕的所有显示单元接收发出的一个或多个变化事件，并从而更新屏幕。此时，如果插件正在通过VC功能进行显示，则VC单元80根据对源树的修改来修改目的树。之后，插件的显示单元通过参考上述经过修改的目的树来对屏幕进行修改。

例如，当通过专用插件来实现源显示和树型视图显示时，源显示插件和树显示插件通过直接参考源树而不是利用目的树来实现它们的显

示。在这种情况下，当在屏幕的任何区域中完成编辑时，源显示插件和树显示插件通过参考修改后的源树来更新屏幕。同样，负责显示区域96的HTML单元50通过参考已根据对源树的修改而做了修改的目的树来更新屏幕。

源显示和树显示也可通过使用VC功能而实现。也就是说，例如，如果HTML被用于源和树型结构的布局，则XML文档可映射为HTML以通过HTML单元50来显示。在这种情况下，将创建具有源格式、树格式、表格式的三个目的树。如果在屏幕上的三个区域的任意一个中进行编辑，则VC单元80对源树进行修改，并在之后分别对源格式、树格式、表格式的三个目的树进行修改。然后，HTML单元50通过参考这些目的树来更新屏幕的三个区域。

以这种方式，在单个屏幕上以多种显示格式显示文档，从而提高了用户的便利性。例如，用户能够利用表90等以视觉上易于理解的格式显示和编辑文档，同时通过源显示或树显示来理解文档的分级结构。在上述实施例中，单个屏幕被划分为多个显示格式，它们被同时显示。但是，也可在单个屏幕上显示单个显示格式，从而可通过用户指令来切换显示格式。在这种情况下，主控单元22从用户处接收用于切换显示格式的请求，并随后命令各插件进行显示切换。

图9示出了由文档处理装置20编辑的XML文档的另一例。在图9所示的XML文档中，XHTML文档被嵌入SVG文档的“foreignObject”标签中，而该XHTML文档包含以MathML描述的公式。在这种情况下，编辑单元24通过参考命名空间而将描绘任务分配或指派给适当的显示系统。在图9所示的实施例中，编辑单元24首先使SVG单元60描绘矩形，然后使HTML单元50描绘XHTML文档。此外，编辑单元24使MathML单元（未示出）描绘公式。以这种方式，包含多个词汇的复合文档被适当地显示。图10示出了显示结果。

在对文档进行编辑期间，待显示的菜单可根据光标（キャリッジ）的位置被切换。也就是说，当光标位于显示SVG文档的区域中时，显示SVG单元60提供的菜单、或用于映射SVG文档的定义文件中定义的命令。当光标位于显示XHTML文档的区域中时，显示HTML单元50提供的

菜单、或用于映射XHTML文档的定义文件中定义的命令。因此，可根据编辑位置提供适当的用户界面。

如果在复合文档中不存在与某词汇相符的适当插件或映射定义，则以该词汇描述的部分可以源或树格式显示。在传统实践中，当要打开在某个文档中嵌有其它文档的复合文档时，如果没有安装能够显示该嵌入文档的应用程序，则它们的内容不能显示。但是，根据本前提技术，即使不存在用于显示的应用程序，也可以将由文本数据组成的XML文档显示为源或树格式，从而能够确定其内容。这是基于文本的XML文档或类似文档的一个特征。

以基于文本的语言来描述数据的另一个有益方面例如在于，在同一文档中以其它词汇描述的部分的数据可被该复合文档中以某个词汇描述的另一文档所参考。此外，当在该文档中进行搜索时，嵌入SVG等图片中的字符串也可作为被搜索的对象。

在以某个词汇描述的文档中，可使用其它词汇的标签。虽然该XML文档通常并不有效，但只要它结构良好（well-formed），就可作为有效的XML文档而被处理。在这种情况下，被插入的属于其它词汇的标签可使用定义文件来进行映射。例如，在XHTML文档中，可使用诸如“重要”和“最重要”的标签以通过强调的方式来显示这些标签周围的部分，或者可将这些标签按重要性的顺序来排序以进行相应显示。

当用户在图10所示的编辑屏幕上对文档进行编辑时，负责对被编辑的部分进行处理的插件或VC单元80对源树进行修改。能够为源树中的各个节点注册对于变化事件的监听器。通常，与各个节点所属的词汇相符的插件的显示单元或VC单元80被注册为监听器。当源树被修改时，DOM提供单元32从被修改的节点向较高层次探索。如果存在注册的监听器，则DOM提供单元32向该监听器发出变化事件。例如，参考如图9中所示的文档，如果位于<html>节点下方的节点被修改，那么该变化事件被通报给被注册为<html>节点的监听器的HTML单元50。在同一时刻，该变化事件被通报给被注册为位于<html>节点上方的<svg>节点中的监听器的SVG单元60。此时，HTML单元50通过参考被修改的源树而更新显示。由于属于SVG单元60本身的词汇的节点并未被修改，因此SVG单

元60可忽视该变化事件。

根据编辑的内容，可以随着HTML单元50对显示进行的修改来改变总体布局。在这种情况下，对于各插件的各个显示区域的布局将由管理屏幕布局的组件（例如，负责显示最高节点的插件）来更新。例如，当由HTML单元50显示的区域较之以前变大时，HTML单元50首先描绘HTML单元50本身所负责的区域，然后确定显示区域的大小。然后，显示区域的大小被通报给管理屏幕布局的组件，以请求对布局进行更新。负责屏幕布局的组件一收到该通知便为各个插件重新布置显示区域。因此，被编辑的部分的显示被适当更新，且总体屏幕布局被更新。

接着，对实现前提技术的文档处理装置20的功能构成进一步进行详细说明。在下面的说明中，使用英文名称对类名等进行描述。

A. 概要

由于因特网的出现，由用户处理、管理的文档的数量大致是成指数函数形式在增加。形成因特网核心的Web（World Wide Web: 万维网）成了接受这些文档数据的大容器。除了文档以外，Web还提供用于这些文档的信息检索系统。这些文档通常是由标记语言描述的。作为标记语言的简单且常用的例子，HTML（HyperText Markup Language: 超文本标记语言）是其中的一个。这样的文档还包括有指向记录在Web的其它位置的其他文档的链接。XML（eXtensible Markup Language: 可扩展标记语言）是更高度普及的标记语言。用于访问和阅览Web文档的简单的浏览器是由像Java（注册商标）那样的面向对象的编程语言开发的。

由标记语言描述的文档通常在浏览器或者其他应用程序中以树数据结构的形式来表现。该结构相当于对文档进行语法分析结果的树。DOM（Document Object Model: 文档目标模型）是用于表述和操作文档的、众所周知的基于树的数据结构模型。DOM提供表述含有HTML或XML文档等的对象集合。DOM包括两个基本组件，即表述文档内的组件的对象是如何连接在一起的标准模型，以及用于对这些对象进行访问或者操作的标准接口。

应用程序的开发者能够支持DOM作为与其自身的数据结构或者API（Application Program Interface: 应用程序接口）的接口。另一方面，

创建文档的应用程序的开发者可以使用DOM的标准接口而不是其自身的API的特定接口。因此，DOM提供标准接口的能力有效地促进了在各种环境下对文档进行共享，特别是用在Web上。DOM定义有几个版本，根据不同的编程环境及应用程序来使用。

DOM树是基于对应DOM的内容的文档的分级表述。DOM树包括“根”及由根生出的一个以上的“节点”。根有时表述全体文档。中间节点可表述元素，诸如表及表中的行和列。DOM树的“叶”通常表述数据，例如不可进一步分解的文本或者图像。DOM树的各个节点也可以与描述字形、大小、颜色、索引等由节点所表示的元素的参数的属性关联对应。

虽然HTML是一般的用于制作文档所使用的语言，但它是用于格式和布局的语言而不是用于描述数据的语言。表现HTML文档的DOM树的节点是预先定义作为HTML的格式化标签的元素。通常，由于HTML不提供用于数据的详述或者对数据添标签/添标注的功能，因而对HTML文档中的数据的查询格式化在很多场合是困难的。

网络设计者们的目标是，能够把Web上的文档通过软件应用程序进行查询或者处理。与显示方法无关，只要是分级结构化的语言，可以同样查询处理。诸如XML（eXtensible Markup Language）这样的标记语言可以提供这类特征。

众所周知，与HTML相反，XML的优点是文档的设计者可以使用可自由定义的“标签”对数据元素进行标注。这样的数据元素可以分级进行结构化。而且，XML文档可以包含文档类型定义，它描述文档内使用的标签及其相互关系的“语法”。为了定义结构化的XML文档的显示方法，使用CSS（Cascading Style Sheet: 层叠式样式表）或者XSL（XML Style Language: 可扩展标记语言样式语言）。关于DOM、HTML、XML、CSS、XSL及其有关语言特征的附加信息也可以从Web获得。（例如，http://www.w3.org/TR/）

Xpath为了指定XML文档部分的位置，提供共同的语法及语义。作为功能性的示例，是对应于XML文档的DOM树的遍历（移动）。这提供了用于操作XML文档的各种表述所关联的字符串、数字及布尔型字符的基本功能。Xpath不是XML文档表现的语法（例如作为文本来看时是

第几行或者是第几字符这样的语法），而是在DOM树等抽象的、逻辑的结构中操作。通过使用Xpath，例如，可以通过XML文档的DOM树内的分级结构来定位。除了用于寻址的用途之外，XPath还被设计用来测试DOM树中的节点是否与某个模式相匹配。涉及XPath的细节可在http://www.w3.org/TR/XPath中找到。

希望有一种有效的文档处理系统，能够利用XML的已知的优点和特征，处理由标记语言（例如XML）描述的文档，并能够提供一种用于创建和修改文档的友好的用户界面。

在此说明的系统的构成中的一些将通过使用被称作MVC（Model-View-Controller: 模型-视图-控制器）即众所周知的GUI（Graphical User Interface: 图形用户界面）范例进行说明。MVC范例将应用程序或应用程序的接口的一部分分解为三部分，即，模型、视图和控制器。最初开发MVC是为了将传统的输入、处理和输出任务分配到GUI区域。

输入－>处理－>输出

控制器－>模型－>视图

根据MVC范例，外界的建模、对用户的视觉反馈以及用户输入通过模型（M）、视图（V）以及控制器（C）对象来分离从而进行处理。控制器操作以解释诸如用户的鼠标和键盘输入的输入，并将这些用户动作映射为发送至模型和/或视图的命令，以实现适当的改变。模型发挥作用以管理一个以上的数据元素、响应对其状态的询问、并对改变状态的指示作出响应。视图操作以管理显示的矩形区域，并有通过图形和文本的组合将数据显现给用户的功能。

B. 文档处理系统的总配置

文档处理系统的实施例将结合图11-29进行清楚说明。

图11（a）图示了包括将在下面描述的能够提供文档处理系统的、根据现有技术的基础功能的配置结构。装置10包括通过通信路径13连接到存储器12的CPU形式或微处理器等形式的处理器11。存储器12可为当前或将来能够使用的任意ROM和/或RAM形式。通信路径13作为典型的总线而设置。对于鼠标、键盘和语音识别系统等用户输入装置14以及显示

装置15（或者其他用户接口）的输入输出接口16也连接在用于处理器11和存储器12通信的总线上。该结构可以是独立的形式，也可以是由多个终端以及一台以上的服务器连接的网络化的形式，也可以由公知的任何方式来构成。本发明并不受这些组件的配置、它们的集中式或分布式体系结构或者多种组件的通信方式的限制。

此外，本系统以及在此讨论的实施例将作为包括提供各种功能性的若干组件以及子组件的例子来讨论。为了提供期望的功能，这些组件以及子组件可以是硬件和软件的组合，也可以仅由硬件或者仅由软件来实现。而且，硬件、软件及其组合可以由通用计算装置、专用硬件或者它们的组合来实现。因此，组件或者子组件的构成包括执行用于提供组件或者子组件功能的专用软件的通用/专用的计算装置。

图11（b）示出了文档处理系统一个例子的总体方框图。文档在这种文档处理系统中被创建和编辑。这些文档（例如XML等）能够以具有标记语言特征的任何语言来表述。同样，为方便起见，已经创建了用于特定的组件和子组件的术语和标题。但是，这些不应被视作对本文公开的一般教导范围的限制。

文档处理系统可被视为具有两个基本构成。一个构成是“执行环境”101，它是文档处理系统运行的环境。例如，执行环境在对文档进行处理中和管理中不仅支持用户，而且支持系统，提供基本效用和功能。第二构成是“应用程序”102，它由在执行环境中运行的应用程序构成。这些应用程序包括文档本身及其各种表述。

1．执行环境

执行环境101的关键组件是程序调用器（ProgramInvoker，即程序启动单元）103。程序调用器103是用于启动文档处理系统而被访问的基本程序。例如，当用户登录并启动文档处理系统时，程序调用器103被执行。程序调用器103例如可以读取并执行作为插件增加至文档处理系统的功能、启动并运行应用程序、以及读取与文档相关的属性。程序调用器103的功能并不限于此。

当用户希望启动计划在执行环境中运行的应用程序时，程序调用器103找到并启动该应用程序，然后执行该应用程序。

在程序调用器103上联接有插件子系统104、命令子系统105以及资源模块109等若干组件。这些构成将随后进行更详细描述。

a）插件子系统

插件子系统104能够高度灵活和有效地向文档处理系统增加功能。插件子系统104也可以用来修改和去除文档处理系统中存在的功能。此外，可使用插件子系统增加或修改多种功能。例如，可以增加编辑功能（Editlet：编辑单元）以起到支持在屏幕上呈现文档的作用。编辑插件也支持对增加至系统的词汇进行编辑。

插件子系统104包括服务代理（ServiceBroker：服务中介单元）1041。服务代理1041通过管理增加至文档处理系统的插件，作为增加至文档处理系统的服务的中介。

期望的每个功能以服务(Service)1042的形式增加至系统。服务1042的可用类型包括但不限于：应用程序 (Application)服务、区工厂(ZoneFactory，即区生成单元）服务、Editlet（编辑单元）服务、命令工厂（CommandFactory：命令生成单元）服务、连接xpath（ConnectXPath：XPath管理单元）服务、CSS计算（CSSComputation：CSS计算单元）服务等。这些服务及其与系统其余部分的关系将随后详细描述，以更好地理解文档处理系统。

插件和服务之间的关系如下所述。插件是可包括一个以上的服务提供器（ServiceProvider：服务提供单元）的单元。各个服务提供器具有与之相关服务的一个以上的类。例如，通过使用具有适当软件应用程序的单个插件，可将一个以上的服务增加至系统，从而可向系统增加相应的功能。

b）命令子系统

命令子系统105被用来执行与文档的处理相关的命令形式的指令。用户可通过执行一系列指令而执行对文档的操作。例如，通过发出命令形式的指令，用户对文档处理系统中的XML文档相对应的XML的DOM树编辑来处理XML文档。这些命令可利用键盘击键、鼠标点击或其它有效的用户接口动作来输入。有时，也可通过一个命令来执行一个以上的

指令。在这种情况下，这些指令被封装（包含）成一个命令并连续执行。例如，用户希望将错误词语替换为正确词语。在这种情况下，第一个命令可用以在文档中找寻错误词语，第二个命令可用以删除该错误词语，第三个命令可用以输入正确词语。这三个命令可被包装成一个命令。

命令可具有相关功能，例如是下面将要详细记述的“撤消”功能。这些功能可分配给用来创建对象所使用的若干个基类。

命令子系统105的关键组件是命令调用器（CommandInvoker，即命令启动单元）1051，命令调用器1051可操作以有选择性地提供并执行命令。虽然图11（b）中仅示出了一个命令调用器，但也可使用一个以上的命令调用器并可同时执行一个以上的命令。命令调用器1051维护执行命令所需的功能和类。在操作中，要执行的命令1052被置于队列1053中。命令调用器创建连续执行的命令线程。如果在命令调用器中没有正在执行的命令，则由命令调用器1051执行待执行的命令1052。如果命令调用器正在执行命令，则新的命令被置于命令队列1053的末尾。不过，对于各命令调用器1051而言，一次仅执行一个命令。如果指定的命令执行失败，则命令调用器1051将执行异常处理。

可由命令调用器（CommandInvoker）1051执行的命令的类型包括但不限于：可撤消命令(UndoableCommand)1054、异步命令(AsynchronousCommand)1055以及词汇连接命令(VCCommand)1056。可撤消命令1054是那些如果用户希望就能够撤销其结果的命令。可撤消命令的示例为：剪切、复制、插入文本等。在操作中，当用户选择文档的一部分并对该部分应用剪切命令时，如果需要，通过使用可撤消命令，可使被剪切的部分恢复至其未被剪切的状态。

词汇连接命令1056被载入词汇连接描述符（Vocabulary Connection Descriptor: VCD）脚本文件中。词汇连接命令1056是能够由程序员定义的用户指定命令。命令可以是例如用于增加XML片段、删除XML片段、以及设置属性等更抽象命令的组合。这些命令特别涉及对文档进行编辑。

异步命令1055是用于文档的载入或保存等基于系统的命令，与可撤

消命令或词汇连接命令不同，是异步执行。由于异步命令不是可撤消命令，所以其执行的结果不能取消。

c）资源

资源109是向各种类提供某些功能的对象。例如，串资源、图标和缺省键绑定是系统中使用的资源的例子。

2. 应用程序组件

应用程序组件102作为文档处理系统的第二个主要特征，在执行环境101中运行。应用程序组件102包括实际文档和系统内的各种逻辑和物理表述。应用程序组件102还包括用来管理文档所使用的系统组件。应用程序组件102进一步包括用户应用程序（UserApplication）106、应用程序核心108、用户界面107以及核心组件（CoreComponent）110。

a）用户应用程序

用户应用程序106连同程序调用器103一起被载入到系统中。用户应用程序106是将文档、文档的各种表述以及与文档进行交互所需的用户界面结合在一起的“粘合剂”。例如，用户可能希望创建作为项目的一部分的一套文档。载入这些文档后，将创建用于文档的适当表述。用户界面功能作为用户应用程序106的一部分被追加。换言之，用户应用程序106既保持文档的表述也保持文档的各种形式，这些文档的表述使用户能够与形成项目的一部分的文档进行交互。一旦创建了用户应用程序106，每当用户希望与形成项目一部分的文档进行交互时，用户就能够简单地将用户应用程序106载入到执行环境中。

b）核心组件

核心组件（CoreComponent）110提供在多个窗格（Pane）之间共享文档的一种方法。如后述的那样，窗格显示DOM树，并处理屏幕的物理布局。例如，物理屏幕包括在屏幕内的多个描述信息的片断的窗格。实际上，由用户在屏幕上看到的文档可在一个或多个窗格中显示。此外，两个不同的文档可以出现在屏幕上的两个不同窗格中。

如图11（c）所示，屏幕的物理布局也具有树型形式。因此，窗格可被实现为根窗格（RootPane）1084，也可以为子窗格(SubPane)1085。根窗格1084是位于窗格树的根部的窗格，而子窗格1085是除了根窗格1084之外的任何窗格。

核心组件110还提供字体，并充当工具包等用于文档的多个功能性操作的源。由核心组件110执行的任务的一个示例是在多个窗格之间移动鼠标光标。作为被执行的任务的另一个示例，标记某窗格中文档的一部分，并将其复制到包含不同文档的另一窗格上。

c）应用程序核心

如上所述，应用程序组件102由系统处理和管理的文档组成。其中包括系统内的文档的多种逻辑和物理表述。应用程序核心108是应用程序组件102的组件。其功能是保持实际文档及其内的所有数据。应用程序核心108包括文档管理器（DocumentManager: 文档管理单元）1081和文档（Document）1082本身。

文档管理器1081的多个方面将在随后详细描述。文档管理器1081管理文档1082。文档管理器1081也连接至根窗格1084、子窗格1085、剪贴板（ClipBoard）实用程序1087以及快照（SnapShot）实用程序1088。剪贴板实用程序1087提供了保持用户决定增加至剪贴板的部分文档的方法。例如，用户可能希望剪切文档的一部分，并将其保存到新的文档上，用于稍后查看。在这种情况下，剪切的部分被增加至剪贴板。

接着，也对快照实用程序1088进行说明。快照实用程序1088在应用程序从一个状态变为另一状态时，能够记住应用程序的当前状态。

d）用户界面

应用程序组件102的另一组件是用户界面107，其为用户提供一种与系统进行物理交互的方式。例如，用户界面用于用户上载、删除、编辑和管理文档。用户界面包括框架（Frame）1071、菜单栏（MenuBar）1072、状态栏（StatusBar）1073以及URL栏（URLBar）1074。

如通常公知的那样，框架1071被视为物理屏幕的活动区域。菜单栏

1072是包含有为用户提供选项菜单的屏幕区域。状态栏1073是显示应用程序的执行状态的屏幕区域。URL栏1074提供输入用于在互联网上定位的URL地址的区域。

C. 文档管理和相关的数据结构

图12示出了文档管理器1081的细节。其中包括用于在文档处理系统内表述文档的数据结构和组件。为了更好的理解，在这部分描述的组件将利用模型－视图－控制器（MVC）表述范例来进行说明。

文档管理器1081包括文档容器（DocumentContainer）203，文档容器203保持并容纳文档处理系统中的所有文档。联接至文档管理器1081的工具包201提供了由文档管理器1081使用的各种工具。例如，DOM服务（DOMService）是由工具包201提供的能够提供创建、维护和管理与文档相对应的DOM所需的所有功能的工具。作为工具包201提供的另一工具的IO管理器（IOManager: 输入输出管理部）分别管理向系统的输入和来自系统的输出。同样地，流处理器（StreamHandler）是一种处理将比特流形式的文档上载的工具。这些工具形成了工具包201的组件，不过并未在图中特别示出和分配附图标记。

根据MVC范例的表述，模型（M）包括文档的DOM树模型202。如上所述，所有文档均在文档处理系统中被表述为DOM树。此外，文档也形成文档容器203的一部分。

1. DOM模型和区

表述文档的DOM树是具有节点（Node）2021的树。作为DOM树的子集的区（Zone）209包括对应于DOM树内部的一个或多个节点的区域。例如，可能仅有文档的一部分在屏幕上显现，文档可见的这一部分可使用“区”209来表述。利用被称作区工厂（ZoneFactory: 区生成单元）205的插件来创建、操作和处理区。虽然区表述DOM的一部分，但它也可使用一个以上的“命名空间”。如本领域中公知的那样，命名空间是名称的汇集或集合，这些名称在该命名空间中是唯一的。换言之，一个命名空间中不存在相同的名称。

2. “方面”及其与区的关系

方面（Facet）2022是MVC范例的模型（M）部分内的另一组件。它被用来编辑区中的节点。“方面”2022使用不会影响区本身的内容的可执行过程（程序）来组织对于DOM的访问。如下面将说明的那样，这些过程执行与节点相关的重要且有用的操作。

各个节点具有相应的“方面”。通过利用“方面”执行操作来代替直接对DOM中的节点进行操作，DOM的完整性得以确保。否则，如果直接对节点执行操作，那么几个插件可能同时对DOM进行改变，从而造成结果的前后矛盾。

虽然W3C构建的DOM标准定义了用于对节点进行操作的标准接口，实际上，由于对每个词汇或每个节点有特定的操作，优选将这些操作作为API来准备。文档处理系统以“方面”的形式向各节点提供了特有的API，并附加到各节点上。据此，可以提供符合DOM标准的有用的API。此外，通过在标准DOM上后来增加特定的API而不是为每个词汇实现特定的DOM，可对多种词汇进行统一处理，并且可以适当地处理由多种词汇任意组合的混合文档。

词汇是属于命名空间的标签（例如XML标签）的集合。如上所述，命名空间具有唯一的名称（在此为标签）集。词汇表现为表述XML文档的DOM树的子树。这种子树包括区（Zone）。在特定实施例中，标签集的边界由区来限定。区209是利用被称为“区工厂”205的服务创建的。如上所述，区209是对表述文档的DOM树的一部分的内部表述。为了提供对该文档的上述部分的访问，需要逻辑表述。这种逻辑表述通知计算机如何在屏幕上对文档进行逻辑显示。画布（Canvas）210是一种可操作为提供与区相对应的逻辑布局的服务。

另一方面，窗格211是与由画布210提供的与逻辑布局相对应的物理屏幕布局。实际上，用户只是看以字符和图片形式呈现在显示屏上的文档。因此，文档必须通过用于在屏幕上描绘字符和图片的处理来呈现在屏幕上。根据由窗格211提供的物理布局，文档由画布210呈现在屏幕上。

与区209相对应的画布210是利用Editlet 206来创建的。文档的DOM是利用Editlet 206和画布210来编辑的。为了维护原始文档的完整性，Editlet 206和画布服务210使用与区209中的一个或多个节点相对应的“方面”。这些服务并不直接操作区和DOM中的节点。“方面”是利用命令207来操作的。

用户通常通过例如移动屏幕上的光标或键入命令而与屏幕进行交互。提供屏幕的逻辑布局的画布210接收这些光标操作。画布210可使“方面”采取相应的动作。根据这种关系，光标子系统204作为用于文档管理器1081的MVC范例的控制器（C）。画布210也提供处理事件的任务。例如，画布210处理诸如鼠标点击、焦点移动以及由用户发起的类似的操作等事件。

3. 区、“方面”、画布和窗格之间的关系概述

文档处理系统内的文档可从至少四个角度来描述。即：1） 用来保持文档处理系统中的文档的内容和结构的数据结构；2）在不影响文档完整性的同时编辑文档内容的方式；3）文档在屏幕上的逻辑布局；以及4）文档在屏幕上的物理布局。“区”、“方面”、“画布”和“窗格”分别表述与上述四个方面相对应的文档处理系统的组件。

4. 撤消子系统

如上所述，人们希望对文档的任何改变（例如，编辑）都可撤消。例如，用户可执行编辑操作，然后决定撤消该改变。参照图12，撤消子系统212是文档管理器的可进行撤消操作的组件。撤消管理器（UndoManager: 撤销管理单元）2121保存可被用户撤消的、对文档执行的所有操作。例如，用户可执行命令来将文档中的词语替换成另一个词语。之后，该用户可改变主意并决定保留原来的词语。撤消子系统212协助上述操作。撤消管理器2121保存上述可撤消编辑（UndoableEdit: 可撤消编辑）2122的操作。

5. 光标子系统

如上所述，MVC的控制器部分可包括光标子系统204。该光标子系统204接受来自用户的输入。这些输入通常具有命令和/或编辑操作的性质。因此，光标子系统204可被视作是与文档管理器1081相关的MVC范例的控制器（C）部分。

6. 视图

如上所述，画布210表述要显现在屏幕上的文档的逻辑布局。对于XHTML文档实施例而言，画布210可包括盒树（box tree）208，该盒树是文档在屏幕上如何被查看的逻辑表述。上述盒树208可包含在与文档管理器1081有关的MVC范例的视图（V）部分中。

D. 词汇连接

文档处理系统的一个重要特征是提供一种环境，该环境能够把XML文档映射为其他表述来处理，而且对映射后的表述编辑时，将该编辑反映到源XML文档中，同时保持该XML文档的完整性。

由标记语言描述的文档，例如XML文档，基于通过文档类型定义的词汇创建。词汇则是一组标签集。由于词汇可以任意定义，这就使得词汇的数量可能是无限的。但是，为多个可能的词汇中的每一个都提供专用的单独处理和管理环境是不切实际的。词汇连接提供解决这种问题的一种方式。

例如，文档可以利用两种以上的标记语言来表述。这些文档例如可以是XHTML（可扩展超文本标记语言）、SVG（可缩放矢量图形）、MathML（数学标记语言）或其他的标记语言。换句话说，标记语言可以视为和XML中的词汇和标签集相同。

词汇使用词汇插件来实现。在文档处理系统中，通过将以插件不可用的词汇所描述的文档映射为插件可用的另一词汇来显示。因此，对于以未准备有插件的词汇描述的文档仍然是可以正确显示。

词汇连接包括获取定义文件、和根据获取的定义文件在两个不同的词汇之间进行映射的能力。用某种词汇描述的文档能够映射为另外的词汇。因此，词汇连接提供通过与文档已被映射成的词汇相对应的显示和

编辑插件来显示或编辑文档的能力。

如上所述，各个文档在文档处理系统中被描述为通常具有多个节点的DOM树。“定义文件”为各个节点描述了该节点与其他节点之间的连接。规定了是否可以对各个节点的元素值和属性值进行编辑。还可以描述使用节点的元素值和属性值的运算表达式。

利用映射特征，可以创建采用定义文件的目的DOM树。由此建立并维护源DOM树和目的DOM树之间的关系。词汇连接监视源DOM树和目的DOM树之间的连接。在从用户接收到编辑指令后，词汇连接修改源DOM树中的相关节点。发出表示已经修改了源DOM树的“变化事件”，并且相应地修改目的DOM树。

通过使用词汇连接，使得仅有少量用户知道的相对次要的词汇可以被转换为其他主要的词汇。因此，即便是对于那些仅有少量用户使用的次要词汇，也可以准确地显示文档，并提供理想的编辑环境。

因此，作为文档处理系统一部分的词汇连接子系统提供能够对文档进行多种表述的功能。

图13显示了词汇连接（VC: Vocabulary Connection）子系统300。VC子系统300提供了一种维护同一文档的两种可替换表述之间的一致性的方式。例如，两种表述可以是同一文档以两种不同词汇来表现的。如上所述，其中一种可以是源DOM树，而另一种可以是目的DOM树。

1. 词汇连接子系统

利用被称为词汇连接（VocabularyConnection）301的插件在文档处理系统中实现词汇连接子系统300的功能。表述文档的各词汇305都需要相应的插件。例如，如果文档的一部分以HTML表述，而其他部分以SVG表述，则需要与HTML和SVG分别对应的词汇插件。

词汇连接插件301为区209或窗格211创建与适当词汇305的文档相对应的适当的词汇连接画布（VCCanvas）310。使用词汇连接301，根据转换规则，对源DOM树的区209的改变被传递到另一DOM树306的相应区。转换规则以词汇连接描述符（Vocabulary Connection Descriptor: VCD）的形式给出。对于与源DOM和目的DOM之间的这种转换相对应

的各个VCD文件，创建相应的词汇连接管理器（VCManager）302。

2. 连接器（Connector）

连接器304连接源DOM树中的源节点和目的DOM树中的目的节点。连接器304的作用是观察源DOM树中的源节点，和对与源节点相对应的源文档进行的修改（变化）。接着，修改相应的目的DOM树中的节点。连接器304是能够对目的DOM树进行修改的唯一对象。例如，用户可以只对源文档和相应的源DOM树进行修改。之后，连接器304对目的DOM树进行相应的修改。

连接器304被逻辑地链接在一起以形成树结构。连接器304形成的树被称为连接器树（ConnectorTree）。连接器304通过一种服务（Service）而创建，该服务被称为连接器工厂（ConnectorFactory: 连接器生成单元）303的服务。连接器工厂303从源文档创建连接器304，并将其链接起来以形成连接器树。词汇连接管理器302维护连接器工厂303。

如上所述，词汇是命名空间中的标签集。如图所示，通过词汇连接301为文档创建词汇305。这通过分析文档文件以及为源DOM和目的DOM之间的映射创建适当的词汇连接管理器302来实现。此外，在创建连接器的连接器工厂303、创建区209的区工厂(ZoneFactory)205和创建与区内节点相对应的画布的Editlet 206之间建立适当的关联。当用户从系统中丢弃或删除文档时，对应的词汇连接管理器302将被删除。

词汇305创建词汇连接画布310。此外，连接器304和目的DOM树306被相应地创建。

源DOM和画布分别对应于模型（M）和视图（V）。然而，仅当目标词汇能够在屏幕上呈现时，这种表现才有意义。这种呈现通过词汇插件来进行。针对主要的词汇（例如XHTML、SVG和MathML）提供词汇插件。词汇插件与目标词汇关联使用。它们提供了一种使用词汇连接描述符在词汇之间进行映射的方式。

仅在目标词汇可被映射并具有预先定义的屏幕呈现方式时，这种映射才有意义。这种呈现方式例如是由诸如W3C组织定义的XHTML等之

类的标准规格。

在需要词汇连接时，使用词汇连接画布（VCCanvas）。在这种情况下，由于不能够为源直接创建视图，因此，不创建源的画布。在这种情况下，使用连接器树来创建词汇连接画布。这种词汇连接画布仅仅处理事件转换，而并不会有助于将文档呈现在屏幕上。

3. 目的区（DestinationZone）、窗格(Pane) 以及画布(Canvas)

如上所述，词汇连接子系统的目的在于同时创建并维护对同一文档的两种表述。第二表述也是DOM树形式，先前作为目的DOM树已被说明。为了浏览第二种表述的文档，需要目的区、画布和窗格。

在创建词汇连接画布后，将创建相应的目的窗格（DestinationPane）307。此外，相关的目的画布（DestinationCanvas）308和相应的盒树(BoxTree)309被创建。同样，词汇连接画布310还与源文档的窗格211和区209关联。

目的画布308提供了文档的第二种表述方式的逻辑布局。具体地，目的画布308提供了用户界面功能，例如光标和选择，用于以目的表述的方式呈现文档。在目的画布308中发生的事件被提供到连接器。目的画布308向连接器304通知鼠标事件、键盘事件、拖动和放置事件、以及通知文档的目的（第二种）表述的词汇的特有事件。

4. 词汇连接命令子系统

作为词汇连接（VC）子系统300的一部分是词汇连接（VC）命令子系统313。词汇连接命令子系统313创建词汇连接命令（VCCommand）315，词汇连接命令315用来执行与词汇连接子系统300相关的指令。可通过内建的命令模板（CommandTemplate）318来创建词汇连接命令，和/或可通过在脚本子系统314中使用脚本语言从无到有地创建命令而创建词汇连接命令。

命令模板中包括，例如“If”命令模板、“When”命令模板、“Insert fragment”命令模板等。这些模板被用来创建词汇连接命令。

5. XPath子系统

XPath子系统316是文档处理系统的一个重要组件，它有助于实现词汇连接。连接器304通常包括XPath信息。如上所述，词汇连接的任务之一是将源DOM树的变化反映到目的DOM树中。XPath信息包括一个或多个XPath表述，用来确定对改变/修改加以监视的源DOM树的子集。

6. 源DOM树、目的DOM树和连接器树（ConnectorTree）的概述

源DOM树是对转换为另一种词汇之前的词汇表述的文档进行表述的DOM树或区。在源DOM树中的节点被称为源节点。

另一方面，如已在前面结合词汇连接描述的，目的DOM树是通过映射转换后以不同词汇表述的文档的DOM树或区。目的DOM树中的节点被称为目的节点。

连接器树是基于用来表述源节点和目的节点之间的对应关系的连接器的分级表述。连接器监视源节点和对源文档进行的修改，修改目的DOM树。连接器是被允许修改目的DOM树的唯一对象。

E. 文档处理系统中的事件流

为了能够使用，程序必需对来自用户的命令进行响应。事件是一种描述和执行用户对程序实施的动作的方法。许多高级语言例如Java（注册商标）依靠描述用户动作的事件。在现有技术中，程序需要主动收集信息以理解用户操作和由程序自己执行用户的操作。这意味着，例如，在对程序自身初始化后，为了在用户对屏幕、键盘和鼠标等执行了任何操作时进行适当的处理，进入反复确认用户操作的循环。然而，这种处理难以操控。此外，这种处理在等候用户作某些事情时，还需要执行循环的程序，从而消耗了CPU周期。

许多语言通过包含不同的范例来解决这些问题。其中的一个范例即事件驱动程序构成了所有现代的视图系统的基础。在这种范例中，所有的用户动作属于被称为“事件”的抽象现象的集合。事件足够详细地描述了特定的用户动作。程序不是主动地收集用户创建的事件，而是在要监视的事件发生时，由系统通知程序。以这种方式处理用户交互的程序被

称为“事件驱动”。

在多数场合，使用“事件（Event）”类来进行处理，其中事件类获取所有用户创建事件的基本特性。

文档处理系统定义和使用其自身的事件以及处理这些事件的方式。有几种类型的事件被使用。例如，鼠标事件是由用户的鼠标操作引起的事件。与鼠标有关的用户操作由画布210传递到鼠标事件。因此，画布可以被认为是用户与系统交互的最前沿。如果需要，作为最前沿的画布将把其与事件有关的内容传递到其下（子）级。

另一方面，按键事件从画布210产生。按键事件具有瞬时的焦点。即，按键事件总是涉及操作。输入到画布210的按键事件接着被传递到其上级（父）。键盘输入通过能够处理字符串插入的不同事件而被处理。处理字符串插入的事件将在使用键盘插入字符时发生。其他的“事件”例如包括以与拖动事件、放置事件和鼠标事件相似的方式处理的其他事件。

1. 在词汇连接之外的事件处理

使用事件线程对事件进行传递。在接收到事件后，画布210改变其状态。如果需要，画布210将命令（Command）1052记入到命令队列（CommandQueue）1053。

2. 在词汇连接之内的事件处理

通过使用词汇连接插件301，作为目的画布一例的XHTML画布（XHTMLCanvas）1106接收现有的事件，例如鼠标事件、键盘事件、拖动和放置事件、以及词汇中的特有事件。这些事件接着被通知到连接器304。更具体地说，如图21（b）所示，词汇连接插件301内的事件流经过源窗格（SourcePane）1103、词汇画布1104、目的窗格1105、作为目的画布的实施例的目的画布1106、目的DOM树和连接器树。

F. 程序调用器（ProgramInvoker）及其与其他组件之间的关系

在图14（a）中更加详细地显示了程序调用器103及其与其他组件之

间的关系。程序调用器103是在执行环境中被执行以启动文档处理系统的基本程序。如图11（b）及图11（c）所示，用户应用程序（UserApplication）106、服务代理(ServiceBroker)1041、命令调用器(CommandInvoker)1051和资源(Resource)109都被联接到程序调用器103。如前所述，应用程序102是在执行环境中运行的组件。同样，服务代理1041管理向系统增加各种功能的插件。另一方面，命令调用器1051执行用户提供的指令，维护用来执行命令的类和函数。

1. 插件和服务

下面将参照图14（b）详细描述服务代理1041。如上所述，服务代理1041管理向系统增加各种功能的插件（及相关服务）。服务1042在最底层，其能够将特征增加到文档处理系统中或者改变文档处理系统的特征。“服务”由两部分构成：服务种类(ServiceCategory)401和服务提供器(ServiceProvider)402。如图14（c）所示，单个服务种类401可具有多个相关的服务提供器402。每个服务提供器都可操作以执行所有或部分的特定服务种类。另一方面，服务种类401则定义了服务的类型。

服务可分为三种类型：1）“特征服务”，向文档处理系统提供特定特征；2）“应用程序服务”，是由文档处理系统运行的应用程序；3）“环境服务”，提供在整个文档处理系统中需要的特征。

图14（d）中示出了服务的例子。根据应用程序服务的种类（Category），系统实用程序是相应服务提供器的示例。同样，Editlet 206是一个种类，HTML Editlet和SVG Editlet是相应的服务提供器。区工厂205是服务的另一种，并具有相应的服务提供器（未示出）。

已经描述的向文档处理系统增加功能的插件，可以看作是由几个服务提供器402和与其相关的类构成的单元。各个插件都具有在定义文件中记载的从属关系和服务种类401。

2. 程序调用器（ProgramInvoker）和应用程序之间的关系

图14（e）详细显示了程序调用器103和用户应用程序106之间的关系。所需的文档、数据等从存储器中载入。所有需要的插件载入到服务

代理1041。服务代理1041维护并管理所有的插件。可将插件物理地增加到系统，或者可从存储器中载入其功能。在载入插件的内容后，服务代理1041定义相应的插件。接着，相应的用户应用程序106被创建，并被载入执行环境101并联接到程序调用器103。

G. 应用程序服务和环境之间的关系

图15（a）更详细地示出了载入程序调用器103中的应用程序服务的结构。作为命令子系统105组件的命令调用器1051调用或执行程序调用器103内的命令1052。命令1052则是用来在文档处理系统中处理XML等文档和编辑相应的XML DOM树的指令。命令调用器1051维护执行命令1052所需的类和功能。

服务调用器1041也在程序调用器103中执行。用户应用程序106连接到用户界面107和核心组件110。核心组件110提供了一种在所有的窗格中共享文档的方式。核心组件110还提供字体并作为用于窗格的工具包。

图15（b）显示了框架(frame)1071、菜单栏(MenuBar)1072和状态栏(StatusBar)1073之间的关系。

H. 应用程序核心

图16（a）进一步解释了应用程序核心108，其保持所有文档以及作为文档一部分及属于文档的数据。核心组件110联接到管理文档1082的文档管理器(DocumentManager)1081。文档管理器1081是存储到与文档处理系统关联的存储器中的所有文档1082的所有者。

为了便于在屏幕上显示文档，文档管理器1081还连接到根窗格(RootPane)1084。剪贴板(ClipBoard)1087、快照(SnapShot)1088、拖拉和放置(Drag&Drop)601以及重叠(Overlay)602的功能也被联接到核心组件110。

快照1088用来撤消应用程序状态。在用户启动快照1088的功能时，应用程序的当前状态被检测并存储。之后，在应用程序的状态变为另一状态时，所存储的状态的内容被保存下来。在图16（b）中示出了快照1088。在操作中，当应用程序从一个URL移动到另一个时，快照1088记

住先前的状态，从而能够无缝地执行后退和前进操作。

I. 在文档管理器中组织文档

图17（a）更加详细地描述了文档管理器1081以及如何在文档管理器中组织并保存文档。如图11（b）所示，文档管理器1081管理文档1082。在图17（a）显示的实施例中，多个文档中的一个为根文档（RootDocument）701，其他的文档为子文档（SubDocument）702。文档管理器1081连接到根文档701，根文档701则连接到所有的子文档702。

如图12和17（a）所示，文档管理器1081耦合到文档容器203，文档容器203是管理所有文档1082的对象。形成包括DOM服务(DOMService)703和IO管理器（IOManager）704的工具包201（例如，XML工具包）的一部分的工具也提供给文档管理器1081。再参照图17（a），DOM服务703创建基于由文档管理器1081管理的文档的DOM树。各个文档705，不管是根文档701还是子文档702都由相应的文档容器203管理。

图17（b）显示了一组文档A-E是如何以分级结构排列的实施例。文档A为根文档。文档B-D是文档A的子文档。文档E则是文档D的子文档。图17（b）的左侧显示了与此相同文档的分级结构显示在屏幕上的实施例。作为根文档的文档A显示为基础框架。文档A的子文档B-D显示为在基础框架A内的子框架。文档D的子文档E在屏幕上显示为子框架D的子框架。

再参照图17（a），为各个文档容器203创建撤消管理器706（UndoManager: 撤消管理单元）和撤消封装器（Undo Wrapper）707。撤消管理器706和撤消封装器707用来执行可撤消的命令。使用该特征，可以撤消使用编辑操作对文档所作的改变。子文档中的改变也会涉及到根文档。撤消操作考虑到对分级结构内其他文档产生影响的改变，例如，如图7（c）所示，确保维护在分级结构链中的所有文档之间的一致性。

撤消封装器707将与容器203中的子文档相关的撤消对象进行封装，

并将它们和与根文档相关的撤消对象耦合。撤消封装器707收集可撤消编辑接受器（UndoableEditAcceptor: 可撤消编辑接受单元）709可利用的撤消对象。

撤消管理器706和撤消封装器707连接到可撤消编辑接收器709和可撤消编辑源（UndoableEditResource）708。本领域技术人员应该理解，文档705可以是可撤消编辑源708，也可以是可撤消编辑对象的源。

J. 撤消命令和撤消框架

图18（a）和18（b）进一步详细地显示了撤消框架和撤消命令。如图18（a）所示，撤消命令（UndoCommand）801、重做命令（RedoCommand）802和可撤消编辑命令（UndoableEditCommand）803是能够排列在如图11（b）所示的命令调用器1051中的命令，并且被顺序执行。可撤消编辑命令803还进一步联接到可撤消编辑源708和可撤消编辑接受器709。“foo”编辑命令804和“bar”编辑命令805就是可撤消编辑命令的例子。

1. 可撤消编辑命令的执行

图18（b）显示了可撤消编辑命令的执行。首先，假设用户使用编辑命令来编辑文档705。在第一步骤S1，可撤消编辑接受器709被联接到可撤消编辑源708，而可撤消编辑源708为文档705的DOM树。在第二步骤S2，基于由用户发出的命令，使用DOM API对文档705进行编辑。在第三步骤S3，向变化事件监听器通知已经发生了改变。即，在该步骤，监视DOM树中所有改变的监听器检测编辑操作。在第四步骤S4，可撤消的编辑作为撤消管理器706的对象被存储。在第五步骤S5，可撤消编辑接受器709与可撤销编辑源708分开。可撤消编辑源708也可以是文档705本身。

K. 与向系统载入文档有关的步骤

上述子部分描述了系统的各个组件和子组件。下面将描述在使用这些组件时用到的方法。图19（a）显示了如何将文档载入到文档处理系统

中的总体图。在图24-28中，结合特定的例子详细地描述各个步骤。

简言之，文档处理系统从文档中包含的数据构成的二进制数据流创建DOM。为文档中的感兴趣的并属于“区”中的一部分创建顶节点（ApexNode: 顶点节点）。接着确定相应的“窗格”。所确定的窗格从顶节点和物理屏幕表面创建“区”和“画布”。接着，“区”为各个节点创建“方面”，并为它们提供所需信息。画布创建用于呈现DOM树的节点的数据结构。

具体地，文档从存储器901载入。创建文档的DOM树902。创建保持文档的相应文档容器903。接着将文档容器903联接到文档管理器904。DOM树包括根节点，在一些情况下可包括多个次级节点。

这种文档一般既包含文本也包含图形。因此，DOM树例如不仅具有XHTML子树也可以具有SVG子树。XHTML子树具有XHTML顶节点（ApexNode）905。同样，SVG子树具有SVG顶节点906。

在步骤1，将顶节点906联接到窗格907，窗格907是屏幕的逻辑布局。在步骤2，窗格907向作为窗口拥有者（PaneOwner）908的核心组件请求用于顶节点的906区工厂。在步骤3，窗口拥有者908返回区工厂以及用于顶节点906的Editlet。

在步骤4，窗格907创建区909，区909联接至窗格907。在步骤5，区909为各个节点创建“方面”，并联接到相应的节点。在步骤6，窗格907创建画布910。画布910联接到窗格907。在画布910中包括各种命令。在步骤7，画布910则构建用于将文档呈现在屏幕上的数据结构。在XHTML的情况下，这包括盒树结构。

1. 用于区的MVC

图19（b）使用MVC范例显示了区的结构概要。在这种情况下，由于区和“方面”是与文档相关的输入，模型（M）包括区和“方面”。 由于画布和将文档呈现在屏幕上的数据结构是用户在屏幕上看到的输出，所以视图（V）对应于画布以及数据结构体。由于命令对文档及其各种关系执行控制操作，所以控制（C）包括画布中所包含的命令。

L. 文档的表述

下面将使用图20来描述文档及其各种表述的实施例。在该实施例中使用的文档既包含文本也包含图片。文本使用XHTML表述，而图片用SVG表述。图20详细显示了文档组件以及相应对象的关系的MVC表述。在该示例中，文档1001联接到保持文档1001的文档容器1002。文档通过DOM树1003表述。DOM树包括顶节点1004。

顶节点用阴影圆圈表示。非顶节点用非阴影圆圈表示。用来编辑节点的"方面"用三角形表示，并被联接到相应的节点。由于文档具有文本和图片，所以该文档的DOM树包括XHTML部分和SVG部分。顶节点1004是XHTML子树的最顶部的节点。该节点被联接到XHTML窗格1005，XHTML窗格（XHTMLPane）1005是文档XHTML部分的物理表述的最顶部窗格。顶节点1004还联接到XHTML区(XHTMLZone)1006，其中XHTML区1006是文档的DOM树的一部分。

与节点1004相对应的"方面"还联接到XHTML区1006。XHTML区1006则联接到XHTML窗格1005。XHTML Editlet创建XHTML画布1007，XHTML画布1007是文档的逻辑表述。XHTML画布1007联接到XHTML窗格1005。XHTML画布1007为文档1001的XHTML组件创建盒树(BoxTree)1009。维护和呈现文档的XHTML部分所需的各种命令1008也被增加到XHTML画布1007。

同样，文档的SVG子树的顶节点1010被联接到SVG区(SVGZone)1011，SVG区1011是用于表述文档的SVG组件的文档1001的DOM树的部分。顶节点1010被联接到SVG窗格(SVGPane)1013，SVG窗格1013是文档的SVG部分的物理表述的最顶部窗格。表述文档的SVG部分的逻辑表述的SVG画布(SVGCanvas)1012通过SVGEditlet创建，并被联接到SVG窗格1013。用于将文档的SVG部分呈现在屏幕上的数据结构和命令被联接到SVG画布。例如，如图所示，这种数据结构可包括圆圈、线、矩形等。

对于结合图20描述的文档例表述的一部分，参照图21（a）用已经描述的MVC范例来作进一步描述。图21（a）提供了文档1001的XHTML组件中的MV关系的简化图。模型是用于文档1001的XHTML组件的

XHTML区（XHTMLZone）1101。在XHTML区的树中包括几个节点及其相应的“方面”。相应的XHTML区和窗格是MVC范例的模型（M）部分的一部分。MVC范例的视图（V）部分是用于文档1001的HTML组件的相应的XHTML画布（XHTMLCanvas）1102和盒树。通过画布以及其中所包含的命令，文档的XHTML部分被呈现在屏幕上。键盘和鼠标输入等事件如图所示，向相反方向进行处理。

源窗格（SourcePane）具有附加功能，也就是说，起到DOM保持器的作用。图21（b）提供了在图21（a）中示出的用于文档1001的组件的词汇连接。作为源DOM保持器的源窗格1103包含文档的源DOM树。连接器树通过连接器工厂创建，创建还起到目的DOM的保持器作用的目的窗格（DestinationPane）1105。目的窗格1105以盒树的形式被布置为XHTML目的画布（XHTMLDestinationCanvas）1106。

M. 插件子系统、词汇连接和连接器之间的关系

图22（a）-（c）分别显示了与插件子系统、词汇连接和连接器相关的进一步的细节。插件子系统被用来向文档处理系统增加、或与之交换功能。插件子系统包括服务代理(ServiceBroker)1041。联接到服务代理1041的区工厂服务1201创建用于文档的部分的区。Editlet服务(EditletService)1202也被联接到服务代理1041。Editlet服务1202创建与区中的节点相对应的画布。

区工厂的例子是分别创建XHTML区和SVG区的XHTML区工厂（XHTMLZoneFactory）1211和SVG区工厂（SVGZoneFactory）1212。如上文档示例所述，文档的文本组件可通过创建XHTML区来表述，而图片则可使用SVG区来表述。Editlet服务的示例包括XHTML Editlet 1221和SVGEditlet 1222。

图22（b）进一步详细显示了词汇连接。如上所述，词汇连接是文档处理系统的重要特征，其能够使两种不同方式的文档的表述和显示保持一致。对连接器工厂303加以维护的词汇连接管理器302是词汇连接子系统的一部分。连接器工厂303创建文档的连接器304。如上所述，连接器监视源DOM中的节点，并修改目的DOM中的节点，以维护两种表述

之间的一致性。

模板（Template）317表述用于一些节点的转换规则。词汇连接描述符（VCD）文件是表示一些规则的模板列表，这些规则用于将满足某种路径或规则的元素或元素集合转换为其他的元素。模板317和命令模板318都联接到词汇连接管理器302。词汇连接管理器是管理VCD文件中所有部分的对象。对一个VCD文件创建一个词汇连接管理器对象。

图22（c）提供了有关连接器的进一步的细节。连接器工厂303从源文档中创建连接器。连接器工厂303与词汇、模板和元素模板联接，并分别创建词汇连接器（VocabularyConnector）、模板连接器(TemplateConnector)和元素连接器(ElementConnector)。

词汇连接管理器302维护连接器工厂303，为了创建词汇，读取相应的VCD文件。接着创建连接器工厂303。该连接器工厂303与创建区的区工厂和创建画布的Editlet相关联。

接着，用于目标词汇的Editlet服务创建词汇连接画布。词汇连接画布也创建源DOM树或区中顶点的连接器。根据需要递归地创建子连接器。通过VCD文件中的一组模板创建连接器树。

模板是用于将标记语言的元素转换为其他元素的规则的集合。例如，各个模板与源DOM树或区相匹配。在正确匹配时，创建顶点连接器。例如，模板“A/*/D”匹配所有从节点A开始、在节点D结束的树分支，而不考虑节点A和节点D之间的节点。同样，“//B”对应于所有来自根节点的“B”节点。

N. 与VCD文件相关的连接器树（ConnectorTree）的示例

下面将解释与特定文档相关的处理。名为“MySampleXML”的文档被载入到文档处理系统。图23显示了用于“MySampleXJML”文件的使用词汇连接管理器及连接器工厂树(ConnectorFactoryTree)的VCD脚本的实施例。

在图中显示了脚本文件内的词汇部分、模板部分以及在词汇连接管理器中的相应组件。在标签“vcd:vocabulary”下提供了属性“match”为"sample:root"，“label”为"MySampleXML"，以及“call-template”为

"sampleTemplate"。

在该实施例中，在"MySampleXML"的词汇连接管理器中，词汇包括顶点元素“sample:root”。相应的UI标注为“MySampleXML”。在模板部分，标签为 “vcd:template”，名称为“sample template”。

O. 将文件载入系统的方法的详细实施例

图24－28显示了载入文档 “MySampleXML”的详细描述。在步骤1，如图24（a）所示，文档从存储器1405中载入。DOM服务（DOMService）创建DOM树以及文档管理器1406对应的文档容器1401。文档容器1401联接到文档管理器1406。文档包括XHTML和MySampleXML的子树。XHTML顶节点1403是具有标签”xhtml:html”的XHTML的最顶部的节点。 “MySampleXML”的顶节点1404是具有标签 “sample:root”的 “MySampleXML”的最顶部的节点。

在步骤2，如图24（b）所示，根窗格为文档创建XTML区、“方面”和画布。创建与顶节点1403对应的窗格1407、XHTML区1408、XHTML画布1409和盒树1410。

在步骤3，如图24（c）所示，在发现XHTML区所不理解的标签“sample:root”后，从XHTML画布上的区域创建子窗格。

图25显示了步骤4，在步骤4中，子窗格获取能够处理“sample:root”标签并可创建适当的区的区工厂。这种区工厂在能够执行区工厂的词汇中。区工厂包括 “MySampleXML”中的词汇部分（VocabularySection）的内容。

图26显示了步骤5，在步骤5中，与 “MySampleXML”对应的词汇创建缺省区(DefaultZone)1061。创建相应的Editlet并提供创建相应的画布的子窗格1501。Editlet创建词汇连接画布。接着，词汇连接画布调用包括连接器工厂树的模板部分(TemplateSection)。连接器工厂树创建将构成连接器树的所有的连接器。

图27所示的步骤6中，各个连接器创建目的DOM对象。一些连接器包括XPath信息。XPath信息包括一个以上的XPath表达式，XPath表达式用来确定需要对是否发生了改变/修改加以监测的源DOM树的子集。

在图28所示的步骤7中，词汇从源DOM的窗格形成目的DOM树的目的窗格。这基于源窗格（SourcePane）来完成。目的树的顶节点联接到目的窗格（DestinationPane）以及相应的区。目的窗格创建目的画布（DestinationCanvas）。而且，为目的窗格提供用于以目的格式呈现文档的数据结构和命令，以及用于目的窗格自身的Editlet。

图29（a）显示了只是存在于目的树上的节点上发生时的事件流，而不具有相应的源节点。如鼠标事件和键盘事件等由画布所获取的事件是通过目的树而被传递到元素模板连接器（ElementTemplateConnector）。元素模板连接器不具有相应的源节点，因此被传送的事件并不是对源节点的编辑操作。如果所传送的事件与命令模板（CommandTemplate）中描述的命令相匹配，则元素模板连接器执行相应的动作。如果没有相匹配的命令，则元素模板连接器忽略所传送的事件。

图29（b）显示了在目的树的节点上发生事件的情况下的流（flow），该目的树的节点通过文本连接器（TextOfConnector）与源节点相关联。文本连接器从由源DOM树的XPath规定的节点获取文本节点，并将该文本节点映射为目的DOM树的节点。如鼠标事件和键盘事件等由画布所获取的事件通过目的树而被传送到文本连接器。文本连接器将所传送的事件映射为相应源节点的编辑命令，并将这些命令设置在队列（Queue）1053中。编辑命令是通过“方面”执行的DOM的API调用集合。当执行设置在队列中的命令时，编辑源节点。在编辑源节点时，发出变化事件，并且将对源节点的修改通知到注册为监听器的文本连接器。文本连接器重新建立目的树，从而在相应的目的节点中反映出对源节点的修改。此时，如果包含文本连接器的模板包括控制声明，例如“for each”和“for loop”，则连接器工厂重新评估控制声明。在重建文本连接器后，重建目的树。

（实施例）

图30是用于说明本实施例中的数据处理内容的示意图。服务器装置通过控制使得被对象化的数据能够由多个用户编辑。用户通过客户终端操作服务器装置的数据。首先，服务器装置获取要被编辑/阅览的源数

据。源数据也可以是XML文档文件中包括的数据，也可以是保存在数据库中的数据等任意数据。源数据也可以从多个源文件抽出。服务器装置将这些源数据作为一个源对象进行对象化。

源对象具有作为成员变量的源数据，并提供用于访问这些成员变量的成员函数。用户不是直接访问源文件，而是通过源对象的成员函数访问作为成员变量保持的源数据。当从XML文档文件生成源对象时，还可以提供遵循DOM规格的接口。另外，通过前提技术中叙述的VC功能，源对象的成员变量也可以各种形式层次结构化。例如，还可以准备用于将数据库中包括的源数据结构化的结构定义文件，将这些源数据结构化。由此，生成将层次结构化的源数据设为成员变量的源对象。

源对象由来自客户终端的访问来操作。一种用于将源对象从客户终端操作的方法如下：

1. 下载源对象到客户终端

2. 在客户终端中对源对象编辑操作

3. 将由编辑操作更新的源对象上载到服务器装置，或者将表示编辑内容的编辑信息发送到服务器装置，在服务器装置中更新源对象。

上述的操作是可能实现的。但是，源对象的数据量相对于客户终端的存储器容量来说较大时，将源对象完全部署在客户终端的本地存储器中是不现实的。大多是用户访问源对象的成员函数中一部分就足够了。

因此，为了提高对源对象访问的效率，服务器装置从一个源对象生成多个部分对象。部分对象中包括源数据作为成员变量，并提供用于访问这些源数据的成员函数。但是，部分对象只将源数据中的一部分设为成员变量，因此与源对象相比，数据量特别小。客户终端下载将要访问的作为成员变量的源数据的部分对象，从而代替下载源对象整体。当编辑下载到客户终端中的部分对象时，服务器装置根据其编辑内容更新源对象。

根据这种方法，能够抑制客户终端的存储器使用量。另外，当用户A访问部分对象α时，用户B几乎不影响用户A的访问就能够访问部分对象β。因此，多个用户能够有效地访问源对象。然而，也有多个用户希望同时访问同一部分对象的情况。这种情况下，服务器装置需要调整多

个访问请求。

图31用于说明对某部分对象进行多个访问要求时的控制方法的序列图。在此，将要访问对象的部分对象称作部分对象α。图31示出了从客户终端A、B、C对该部分对象α进行访问请求的情况。首先，当客户终端A向部分对象α请求访问时，对部分对象α生成写标记（Write Token）和读标记（Read Token）的一对标记。以下，将客户终端A的写标记称作“AW”，读标记称作“AR”。对客户终端B、C也以同样方式标记。对每个部分对象都生成标记，当访问不同于部分对象α的部分对象β时，生成新的标记对。

写标记是表示对部分对象的编辑权的标记。读标记是表示对部分对象的阅览权的标记。编辑权是指通过部分对象的成员函数变更成员变量的权利。阅览权是指当部分对象的成员变量变更时实时阅览最新数据的权利。在下文中，将编辑权和阅览权称作“访问权”。读标记和写标记对生成后，其中一个标记保持在服务器装置，另一标记保持在客户终端。在图31中，首先客户终端A请求部分对象α的阅览权。

这种情况下，客户终端A保持AR，而将AW发送到服务器（S10）。客户终端A通过剩下的AR认为保持着阅览权。另一方面，接受AW的服务器装置认为担负着将部分对象α的最新数据通知客户终端A的义务。当服务器装置接受AW时，将部分对象α的数据发送到客户终端A（S12）。接着，客户终端B也请求部分对象α的阅览权。客户终端B将BW发送到服务器，而保持BR（S14）。当服务器装置接受BW时，将部分对象α的数据发送到客户终端B（S16）。

客户终端C请求部分对象α的编辑权。这种情况下，客户终端C将CR发送给服务器装置，并保持CW（S18）。客户终端C通过剩下的CW认为保持着编辑权。另一方面，接受CR的服务器装置认为其担负着从客户终端C接收更新部分对象α的义务。当服务器装置接受CR时，将部分对象α的数据发送到客户终端C（S20）。由此，服务器装置向获取访问权的客户终端发送部分对象α的数据。各客户终端将在本地保持部分对象α的数据。一个部分对象的阅览权提供给多个客户终端，但是提供编辑权的客户终端是一个或者零个。因此，即使未图示的客户终端D重新

请求部分对象α的编辑权，服务器装置也会拒绝该请求。

在获取了编辑权的客户终端C中，当编辑操作本地中保持的部分对象α时，更新部分对象α的内容，即成员变量（S22）。从客户终端C向服务器装置发送表示其编辑内容的编辑信息（S24）。服务器装置更新源对象使得编辑内容反映在源对象的相应部分（S26）。服务器装置向具有阅览权的客户终端A、B发送编辑信息（S28、S30）。客户终端A和B分别根据编辑信息更新本地中保持的部分对象α（S32、S34）。由此，客户终端C中的部分对象α的编辑内容也反映在服务器装置的源对象、以及客户终端A、B的部分对象α中。

在图31中，之后客户终端C放弃编辑权，取而代之请求阅览权。当客户终端C将表示编辑权的CW发送到服务器装置时（S36），服务器装置将CR返回客户终端C（S38）。由此，客户终端C代替放弃编辑权而获取阅览权。由于放弃了部分对象α的编辑权，客户终端B通过发送BR（S40），接受BW（S42）来重新获取编辑权。此外，在客户终端C放弃了对部分对象α的访问权的情况下，客户终端C将CW发送到服务器装置，服务器装置将CW和CR对删除。接着，将参照具体的实施方式说明作为服务器装置和客户终端的这些装置。

图32是数据管理装置的功能框图。数据管理装置3100是作为与图30、图31关联说明的服务器装置工作的装置。这里所说的各模块，硬件能够由以计算机CPU为首的元件、机械装置实现，软件是由计算机程序等实现，但是这里描绘了由这些协同实现的功能模块。因而，这些功能模块能够由硬件、软件的组合以各种形式实现，这对于本领域从业人员是可以理解的。对于下面的图33所示功能框图也相同。

数据管理装置3100包括用户接口处理单元3110、通信单元3120、数据处理单元3130以及数据保存单元3140。用户接口处理单元3110担当着如来自用户的输入处理、对用户的信息显示的用户接口相关的全部处理。通信单元3120担当着与作为客户终端工作的数据处理终端3200之间的通信处理。对于数据处理终端3200将参照图33在下面详述。在本实施例中，以由用户接口处理单元3110提供数据管理装置3100的用户接口服务为例进行说明。作为其他例子，也可以是用户通过因特网操作数据管

理装置3100。

基于通过用户接口处理单元3110的输入操作以及从通信单元3120获取的数据，数据处理单元3130执行各种数据处理。数据处理单元3130还起到用户接口处理单元3110、通信单元3120以及数据保存单元3140之间接口的作用。数据保存单元3140保存各种数据，例如预先准备的各种设定数据，以及从数据处理单元3130接收的数据。

数据保存单元3140包括源数据保持单元3166、对象保持单元3168、访问权信息保持单元3170以及结构化的管理文件保持单元3172。源数据保持单元3166保存作为源对象源的XML文档文件等源数据。对象保持单元3168保存例如源对象和部分对象等的对象数据。访问权信息保持单元3170保持从数据处理终端3200接收的读标记和写标记。结构化的管理文件保持单元3172保存结构化的管理文件。结构化的管理文件是指表示源对象中的节点层次结构。详细情况将后述。

通信单元3120包括访问请求接收单元3142、部分对象发送单元3144、结构化的管理文件发送单元3146、编辑信息接收单元3148以及编辑信息发送单元3150。访问请求接收单元3142从数据处理终端3200接收访问请求信息。这里所说的访问请求信息是用于向部分对象请求编辑权或者阅览权的数据。作为访问请求信息发送的数据是在其中将要访问的部分对象、请求访问的数据处理终端3200、以及读标记或写标记的任何一个标记进行打包的数据。

部分对象发送单元3144向请求访问的数据处理终端3200发送指定了部分对象的数据。结构化的管理文件发送单元3146向各数据处理终端3200发送结构化的管理文件。结构化的管理文件发送单元3146可以向各数据处理终端3200发送结构化的管理文件，而与是否存在访问请求无关。编辑信息接收单元3148从获取了编辑权的数据处理终端3200接收编辑信息。编辑信息发送单元3150向获取了阅览权的数据处理终端3200发送编辑信息。

数据处理单元3130包括对象处理单元3152和访问权管理单元3154。对象处理单元3152执行例如对源数据进行对象化等的面向对象处理。访问权管理单元3154通过读标记和写标记管理每个部分对象的访问权。即

访问权管理单元3154根据从数据处理终端3200接收的读标记和写标记，执行基于访问权的判定处理，例如通过选择确定编辑信息应向那个客户端发送、以及是否执行源对象更新处理。

对象处理单元3152进一步包括源对象生成单元3156、源对象更新单元3158、部分对象生成单元3160以及结构化的管理文件处理单元3162。源对象生成单元3156将源数据结构化来生成源对象。生成的源对象保存在源对象保持单元3168中。部分对象生成单元3160从源对象生成多个部分对象。生成的多个部分对象也保存在对象保持单元3168中。当在具有编辑权的数据处理终端3200中更新部分对象时，编辑信息接收单元3148从该数据处理终端3200接收编辑信息。源对象更新单元3158根据编辑信息更新源对象。根据来自源对象更新单元3158的指示，部分对象生成单元3160也更新相应的部分对象。结构化的管理文件处理单元3162生成表示源对象节点层次结构的结构化的管理文件。当节点层次结构由于源对象更新而变化时，结构化的管理文件处理单元3162更新结构化的管理文件。当更新了结构管理信息时，结构化的管理文件发送单元3146向各数据处理终端3200重新发送更新后的结构化的管理文件。

图33是数据处理终端的功能框图。数据处理终端3200是作为与图30、图31关联而说明的客户终端工作的装置。数据处理终端3200包括用户接口处理单元3210、通信单元3220、数据处理单元3230以及数据保存单元3240。数据处理终端3200的用户接口处理单元3210也担当关于用户接口的全部处理，例如，用户的输入处理、或向用户显示信息的处理。数据处理终端3200担当与作为服务器装置连接的数据管理装置3100之间的通信处理。在本实施例中，以由用户接口处理单元3210提供数据处理终端3200的用户接口服务为例进行说明。作为其他例，也可以是用户通过因特网操作数据处理终端3200。

基于通过用户接口处理单元3210进行的输入操作、或从通信单元3220获取的数据，数据处理单元3230执行各种数据处理。数据处理单元3230还起到用户接口处理单元3210、通信单元3220以及数据保存单元3240之间接口的作用。数据保存单元3240保存各种数据，例如，预先准备的各种设定数据或从数据处理单元3230接收的数据。

数据保存单元3240包括部分对象保持单元3256、结构化的管理文件保持单元3262、访问标记保持单元3264。部分对象保持单元3256保存从数据管理装置3100接收的部分对象。部分对象保持单元3256包括第1保持单元3258和第2保持单元3260。要被编辑或阅览等操作的部分对象保存在第1保持单元3258中。另外，当不对部分对象进行编辑或预览操作时，该部分对象保存在第2保持单元3260中。更具体地说，第1保持单元3258可以由小容量且可高速访问的主存储装置（main memory）实现。另一方面，第2保持单元3260可以由大容量且访问速度相对低速的硬盘等辅助存储装置（auxiliary storage unit）实现。结构化的管理文件保持单元3262保持结构化的管理文件。访问标记保持单元3264对获取了访问权的每个部分对象保持标记。

通信单元3220包括访问请求发送单元3242、部分对象接收单元3244、结构化的管理文件接收单元3246、编辑信息发送单元3248以及编辑信息接收单元3250。访问请求发送单元3242向数据管理装置3100发送访问请求信息。部分对象接收单元3244从数据管理装置3100接收部分对象。结构化的管理文件接收单元3246从数据管理装置3100接收结构化的管理文件。编辑信息发送单元3248从数据管理装置3100接收与要被阅览的部分对象相关的编辑信息。编辑信息接收单元3250将与要被编辑的部分对象相关的编辑信息发送到数据管理装置3100。

数据处理单元3230包括部分对象更新单元3252和访问权管理单元3254。部分对象更新单元3252根据对获取了编辑权的部分对象的编辑操作，更新部分对象保持单元3256的部分对象。另外，当接收了与获取了阅览权的部分对象相关的编辑信息时，根据编辑信息更新部分对象保持单元3256的部分对象。访问权管理单元3254通过标记管理对各部分对象的访问权。在下面，示例从源对象生成部分对象的过程，说明访问源对象时的用户接口。

图34是用于说明从源对象生成部分对象的过程的示意图。在图31中示意性地示出了节点1~16的形成树状结构的源对象。该源对象是包括16个以上节点的巨大对象，但是这里将图31所示的16个节点作为对象说明。当源数据是XML文档文件数据时，由标签确定的元素（element）对

应一个节点。另外，当源数据是规定的数据库中包括的数据时，成为其元素的数据对应一个节点。但是，源对象中的节点和源数据的对应关系可以任意决定。

在图34中，16个节点分配到p1至p8的八个部分对象。例如，部分对象p1是作为成员变量只包括节点1，提供用于操作与节点1相关联的数据的成员函数的对象。另一方面，与p6对应的部分对象包括节点8、9、10、11、24五个节点，提供用于操作这些节点的成员函数。节点和部分对象的对应关系也可以由用户任意规定。例如一个节点也可以对应一个部分对象。

图35是结构化的管理文件的数据结构图。图34所示的各部分对象和各节点的层次结构作为结构化的管理文件而被管理。部分对象栏3270表示部分对象的名。上对象栏3272表示位置处于处于与部分对象栏3270中对应的部分对象之上的部分对象。下对象栏3274表示位置处于与部分对象栏3270中对应的部分对象之下的部分对象。成员节点栏3276表示部分对象中包括的节点名。由此，结构化的管理文件是用于表示不包括每个节点数据内容的源对象层次结构的文件。当源对象的层次结构由节点的追加、删除而变化时，数据管理装置3100的结构化的管理文件处理单元3162更新结构化的管理文件。结构化的管理文件发送单元3146向各数据处理终端3200发送更新后的结构化的管理文件。

此外，部分对象栏3270、上对象栏3272、下对象栏3274也可以表示用于各部分对象的指针。另外，源对象的全部数据没有必要保存在单一的数据管理装置3100中。例如，在源对象数据分散保持在多个数据管理装置3100中的情况下，结构化的管理文件还可以包括与数据管理装置3100有关的信息，即各部分对象实体存在哪里。这种情况下，数据处理终端3200根据请求访问的部分对象，参照结构化的管理文件判断应该向哪个数据管理装置3100发送访问请求信息。

图36是在数据处理终端中选择要被阅访问的数据时的屏幕图。数据处理终端3200的用户接口处理单元3210参照结构化的管理文件显示节点结构显示屏幕3280，以示出源对象中的节点层次结构。用户通过用鼠标点击滚动按钮3284、滚动按钮3282，从而可滚动屏幕来观看整体层次结

构。

与其他部分对象相比，强调显示部分对象p3和部分对象p7。这表示部分对象p3和部分对象p7已经下载完成。换句话说，对于部分对象p3和部分对象p7，获取了访问权。部分对象p3上显示的“W”表示获取了部分对象p3的编辑权。部分对象p7右上显示的“R”表示获取了部分对象p7的阅览权。当部分对象p3管理的节点4被指定要被访问时，部分对象保持单元3256的部分对象p3的数据加载到第1保持单元3258。此时，暂时未被访问的部分对象p7可以保存到第2保持单元3260。

另外，当未保持在本地中的部分对象p4被指定为要被访问时，数据处理终端3200的访问权管理单元3254生成部分对象p4的写标记和读标记对。在用户请求编辑权时，读标记被打包为访问请求信息并被发送到数据管理装置3100。当请求阅览权时，写标记被打包为访问请求信息，并被发送到数据管理装置3100。该数据处理终端3200从数据管理装置3100接收部分对象p4的数据，并将其保存在部分对象保持单元3256的第1保持单元3258中。此时，部分对象p4的显示部分重新变换为强调显示。

在用户放弃对部分对象p3的编辑权来删除访问权本身的情况下，访问请求发送单元3242将访问标记保持单元3264所保持的写标记发送到数据管理装置3100。数据管理装置3100的访问权管理单元3154将已经保持的读标记和重新接收的写标记删除，解除对应于部分对象p3的数据处理终端3200的访问权。另外，数据处理终端3200的部分对象更新单元3252将部分对象p4的数据从部分对象保持单元3256中删去。此时，与部分对象p3相关的显示部分从强调显示改变为普通显示。放弃部分对象的阅览权来删除访问权的情况也与此相同。

以上，在本实施例中说明了由数据管理装置3100和数据处理终端3200构成的数据管理系统。数据处理终端3200的用户查看节点结构显示画面3280的同时，可选择希望访问的数据。当以DOM处理XML文档文件时，按照层次结构顺序浏览节点来确定要处理的节点。在这种被称作遍历树（tree traverse）的处理方法的情况下，当前节点是图36的节点4时，为了处理节点6，经过节点1、2、3、到达节点6。因此，将产生需要临时访问本来不是访问对象的节点1、2、3的必要。但是，根据如本

实施例所示的访问方法，能够直接获取访问对象的部分对象p4。

另外，即使源对象很大，由于是以部分对象为单位进行访问管理，因此能够有效地处理多个用户的访问。

数据处理终端3200根据获取的部分对象是否是要处理的对象，将第1保持单元3258和第2保持单元3260的任何一个作为对其进行保存的装置而进行切换。因此，在访问多个部分对象的情况下，数据处理终端3200可最大限度地利用存储区域。

此外，权利要求书中所述的数据编辑装置和数据阅览装置的各功能，在本实施例中是由数据处理终端3200单体实现的。本领域的技术人员来应该理解，权利要求书中描述的各元素执行的功能，可以由在本实施方式中所示的各功能模块的单体或者它们的协同操作来实现。

以上以实施方式为基础对本发明节进行了描述。该实施方式是示例，本领域技术人员应该理解，在本发明的范围内，可以对它们的各结构元素、各处理流程的组合进行各种改变。

在本实施方式中，说明了处理XML文档的例子，但是对于用其他标记语言，例如SGML、HTML等描述的文档，也能够同样地进行处理。

在对本实施方式的说明中假设了，当对象保持单元3256中不存在包括数据处理终端3200的用户指定要访问的节点的部分对象时，数据处理终端3200向数据管理装置3100请求相应的部分对象。作为变形例，数据管理装置3100不仅可以提供与请求要访问的部分对象相关的访问权，而且可以提供与要访问对象周边的部分对象（例如，上部分对象或下部分对象）相关的访问权。此时，共同下载要访问的多个部分对象。根据这种方式，不仅可以获取当前请求要访问的部分对象，而且还能够获取与认为将来有可能访问的部分对象有关的访问权，因此将容易抑制通信处理次数。

另外，在本实施例中的说明中假设了数据管理装置3100从源对象预先生成部分对象。作为变形例，数据管理装置3100也能够以数据处理终端3200的访问请求为契机，从源对象的相应部分适当生成部分对象。

产业上利用的可能性

根据本发明，多个用户容易有效地操作从源数据生成的对象

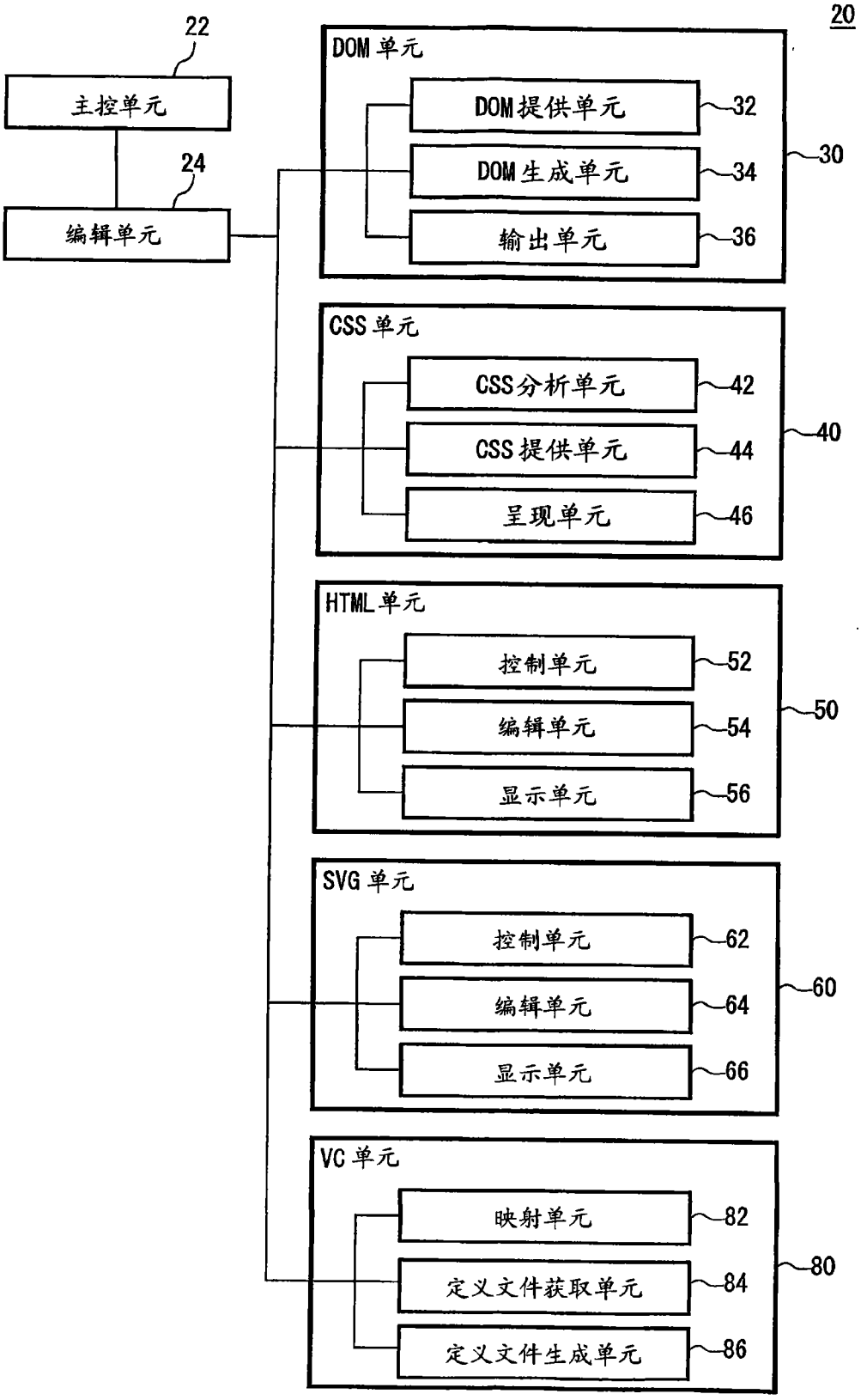

20
22
主控单元
24
编辑单元
DOM 单元
DOM提供单元
32
DOM生成单元
34
输出单元
36
30
CSS 单元
CSS分析单元
42
CSS提供单元
44
呈现单元
46
40
HTML单元
控制单元
52
编辑单元
54
显示单元
56
50
SVG 单元
控制单元
62
编辑单元
64
显示单元
66
60
VC 单元
映射单元
82
定义文件获取单元
84
定义文件生成单元
86
80

图 1

```
<?xml version="1.0" ?>

<?com.xfytec.vocabulary-connection href="records.vcd" ?>
<成績 xmlns="http://xmlns.xfytec.com/sample/records">
  <生徒 名前="A">
    <国語>90</国語>
    <数学>50</数学>
    <理科>75</理科>
    <社会>60</社会>
  </生徒>
  <生徒 名前="B">
    <国語>45</国語>
    <数学>60</数学>
    <理科>55</理科>
    <社会>50</社会>
  </生徒>
  <生徒 名前="C">
    <国語>55</国語>
    <数学>45</数学>
    <理科>95</理科>
    <社会>40</社会>
  </生徒>
  <生徒 名前="D">
    <国語>25</国語>
    <数学>35</数学>
    <理科>40</理科>
    <社会>15</社会>
  </生徒>
</成績>
```

图 2

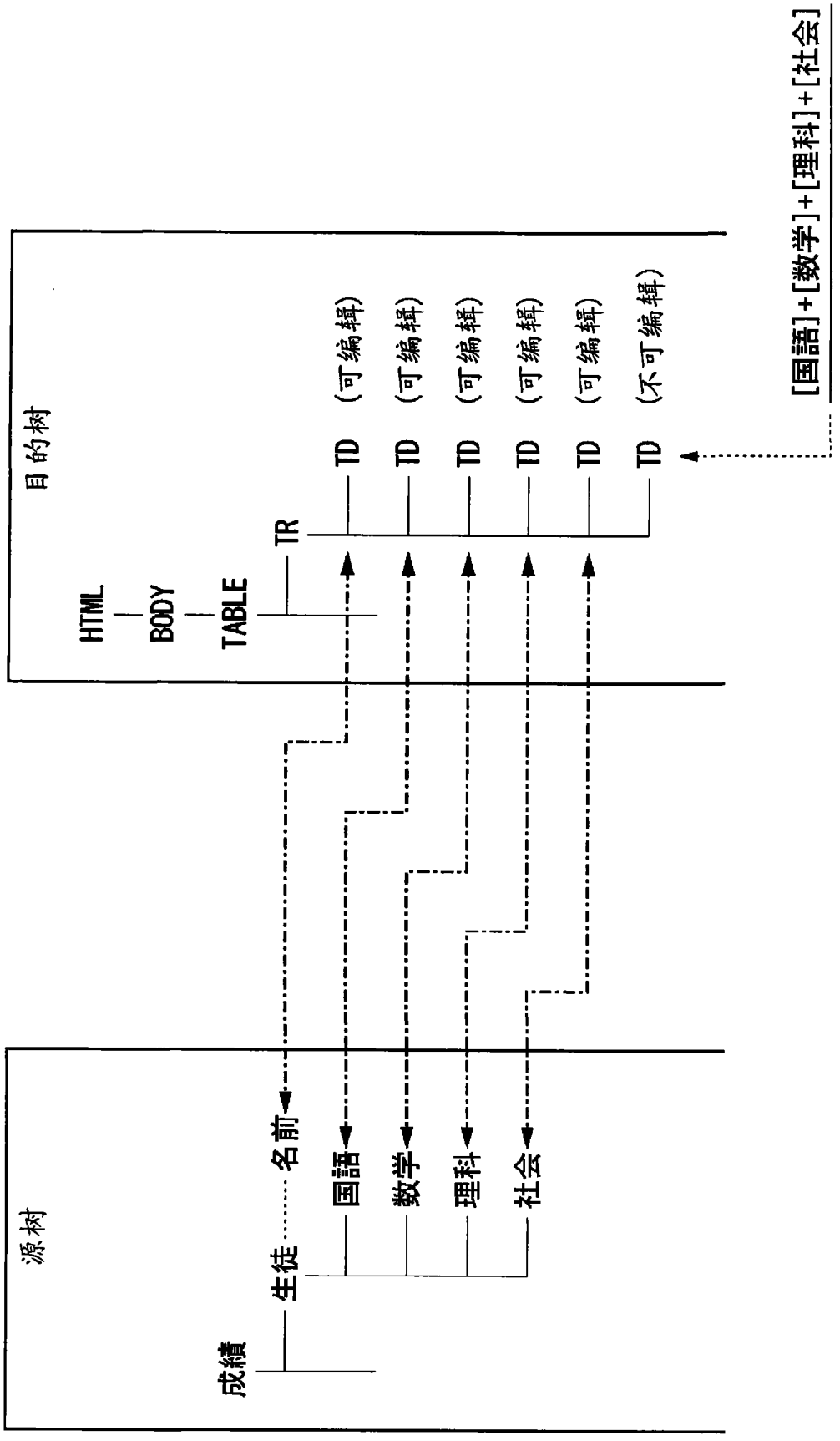

源树
成績
生徒
名前
国語
数学
理科
社会
目的树
HTML
BODY
TABLE
TR
TD (可编辑)
TD (可编辑)
TD (可编辑)
TD (可编辑)
TD (可编辑)
TD (不可编辑)
[国語]+[数学]+[理科]+[社会]
4

图 3

```
<?xml version="1.0"?>

<vc:vcd xmlns:vc="http://xmlns.xfytec.com/vcd"
         xmlns:src="http://xmlns.xfytec.com/sample/records"
         xmlns="http://www.w3.org/1999/xhtml"
         version="1.0">

<!-- Commands -->
<vc:command name="生徒の追加">
  <vc:insert-fragment
      target="ancestor-or-self::src:生徒"
      position="after">
    <src:生徒/>
  </vc:insert-fragment>
</vc:command>
<vc:command name="生徒の削除">
  <vc:delete-fragment target="ancestor-or-self::src:生徒" />
</vc:command>

<!-- Templates -->
<vc:vc-template match="src:成績" name="成績表" >

  <vc:ui command="生徒の追加">
    <vc:mount-point>
      /MenuBar/成績表/生徒の追加
    </vc:mount-point>
  </vc:ui>
  <vc:ui command="生徒の削除">
    <vc:mount-point>
      /MenuBar/成績表/生徒の削除
    </vc:mount-point>
  </vc:ui>

  <html>
    <head>
      <title>成績表</title>
      <style>
        td,th {
          text-align:center;
          border-right:solid black 1px;
          border-bottom:solid black 1px;
          border-top:none 0px;
          border-left:none 0px;
        }
        table{
          border-top:solid black 2px;
          border-left:solid black 2px;
          border-right:solid black 1px;
          border-bottom:solid black 1px;
          border-spacing:0px;
        }
```

图 4(a)

```
        tr{
          border:none;
        }
        .data{
          padding:0.2em 0.5em;
        }
      </style>
    </head>
    <body>
      <h1>成績一覧</h1>
      <table>
        <tr><th><div class="data">名前</div></th>
        <th></th>
        <th><div class="data">国語</div></th>
        <th><div class="data">数学</div></th>
        <th><div class="data">理科</div></th>
        <th><div class="data">社会</div></th>
        <th></th>
        <th><div class="data">平均</div></th></tr>
        <vc:apply-templates select="src:生徒" />
      </table>
    </body>
  </html>
</vc:vc-template>

<vc:template match="src:生徒">
  <tr>
    <td><div class="data">
      <vc:text-of select="@名前" fallback="名無し"/>
    </div></td>
    <td></td>
    <td><div class="data">
      <vc:text-of select="src:国語" fallback="0" type="vc:integer" />
    </div></td>
    <td><div class="data">
      <vc:text-of select="src:数学" fallback="0" type="vc:integer" />
    </div></td>
    <td><div class="data">
      <vc:text-of select="src:理科" fallback="0" type="vc:integer" />
    </div></td>
    <td><div class="data">
      <vc:text-of select="src:社会" fallback="0" type="vc:integer" />
    </div></td>
    <td></td>
    <td><div class="data">
      <vc:value-of
          select="(src:国語 + src:数学 + src:理科 + src:社会) div 4" />
    </div></td>
  </tr>
</vc:template>

</vc:vcd>
```

图 4(b)

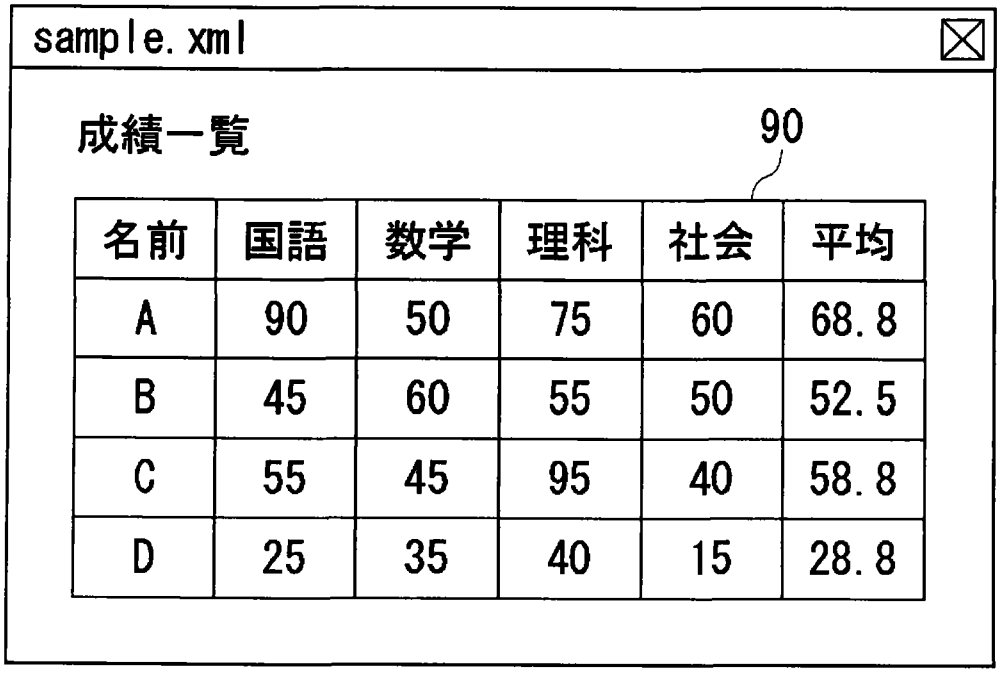

sample.xml
成績一覧
90
名前 国語 数学 理科 社会 平均
A 90 50 75 60 68.8
B 45 60 55 50 52.5
C 55 45 95 40 58.8
D 25 35 40 15 28.8

图 5

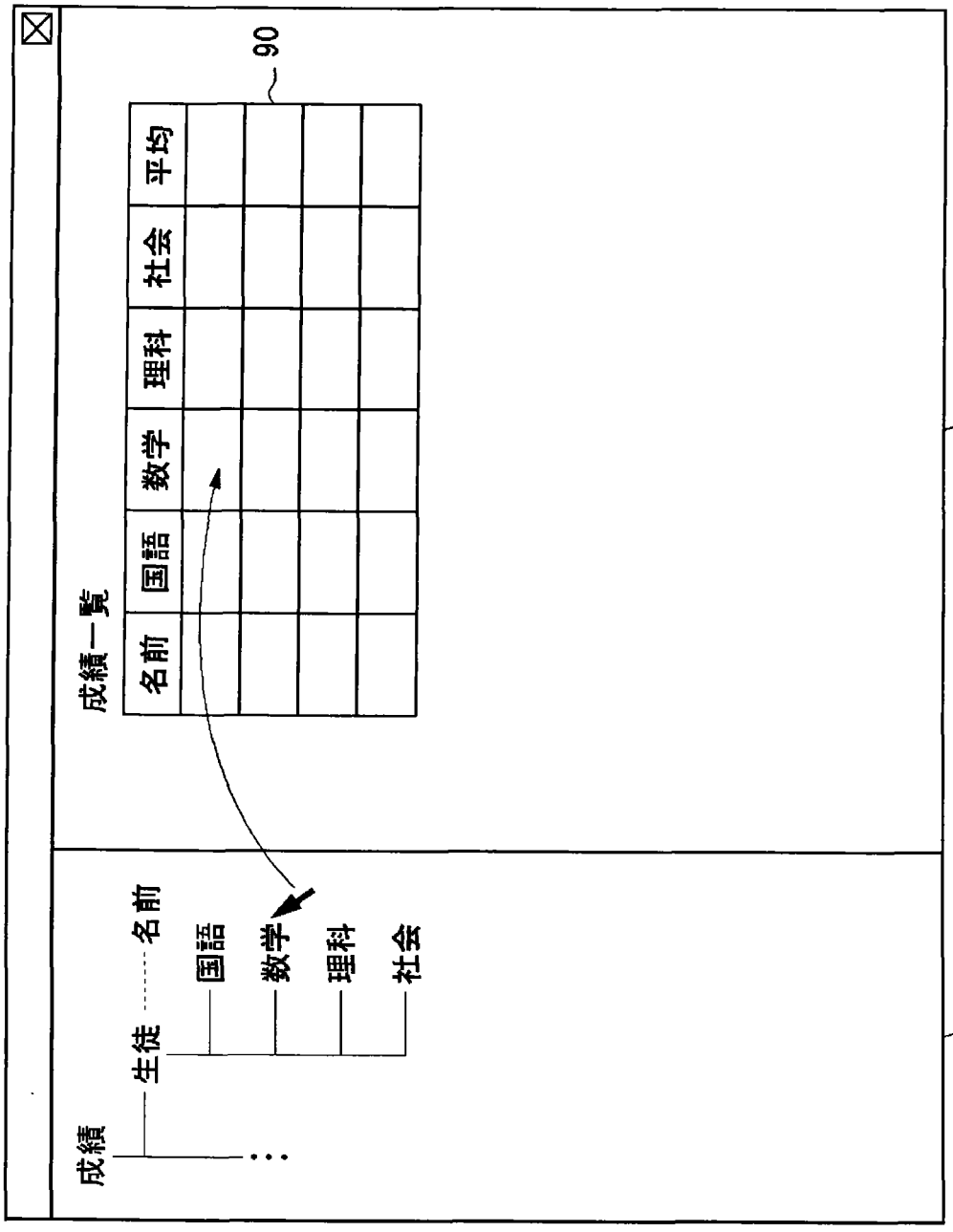


图 6

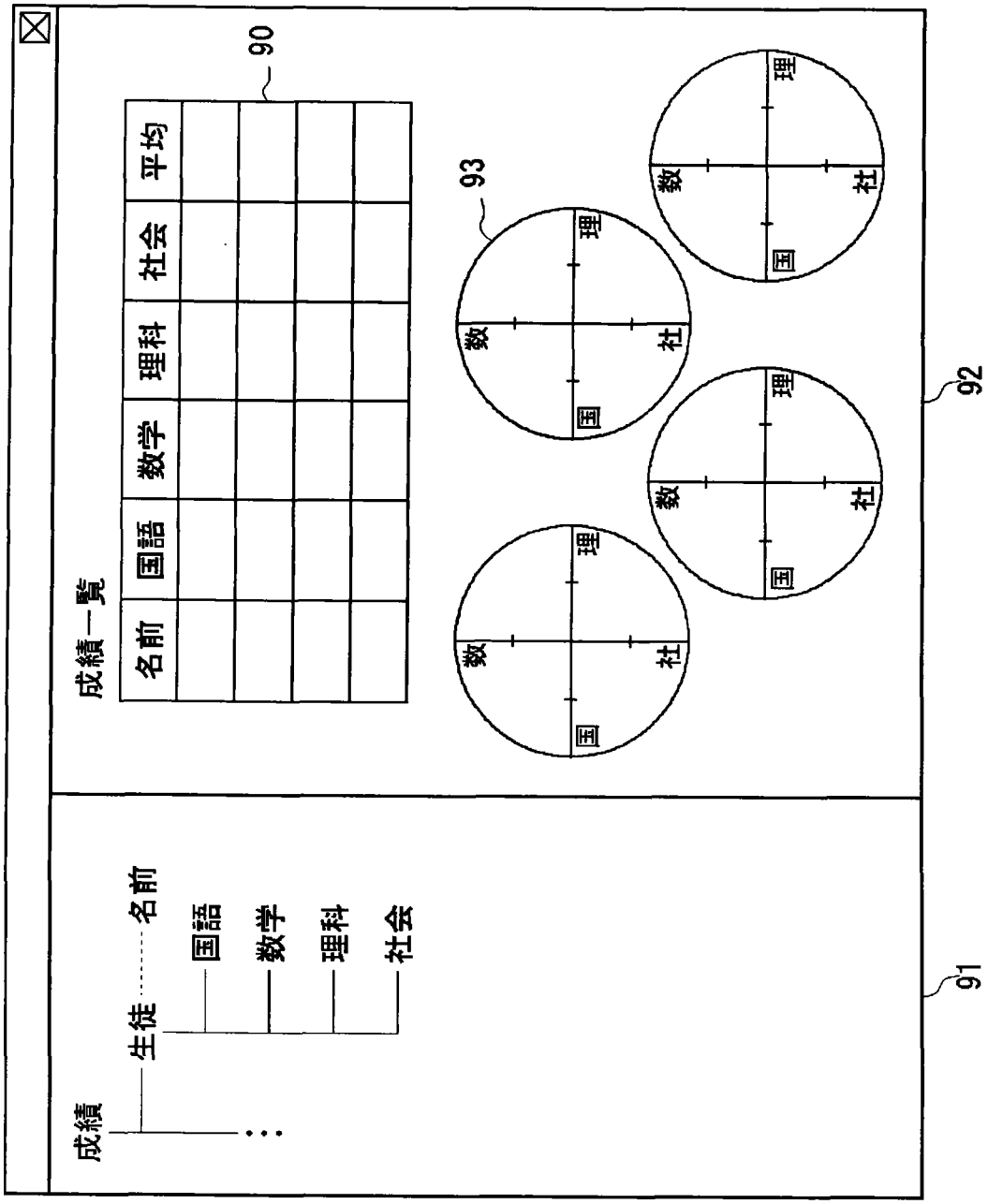


图 7

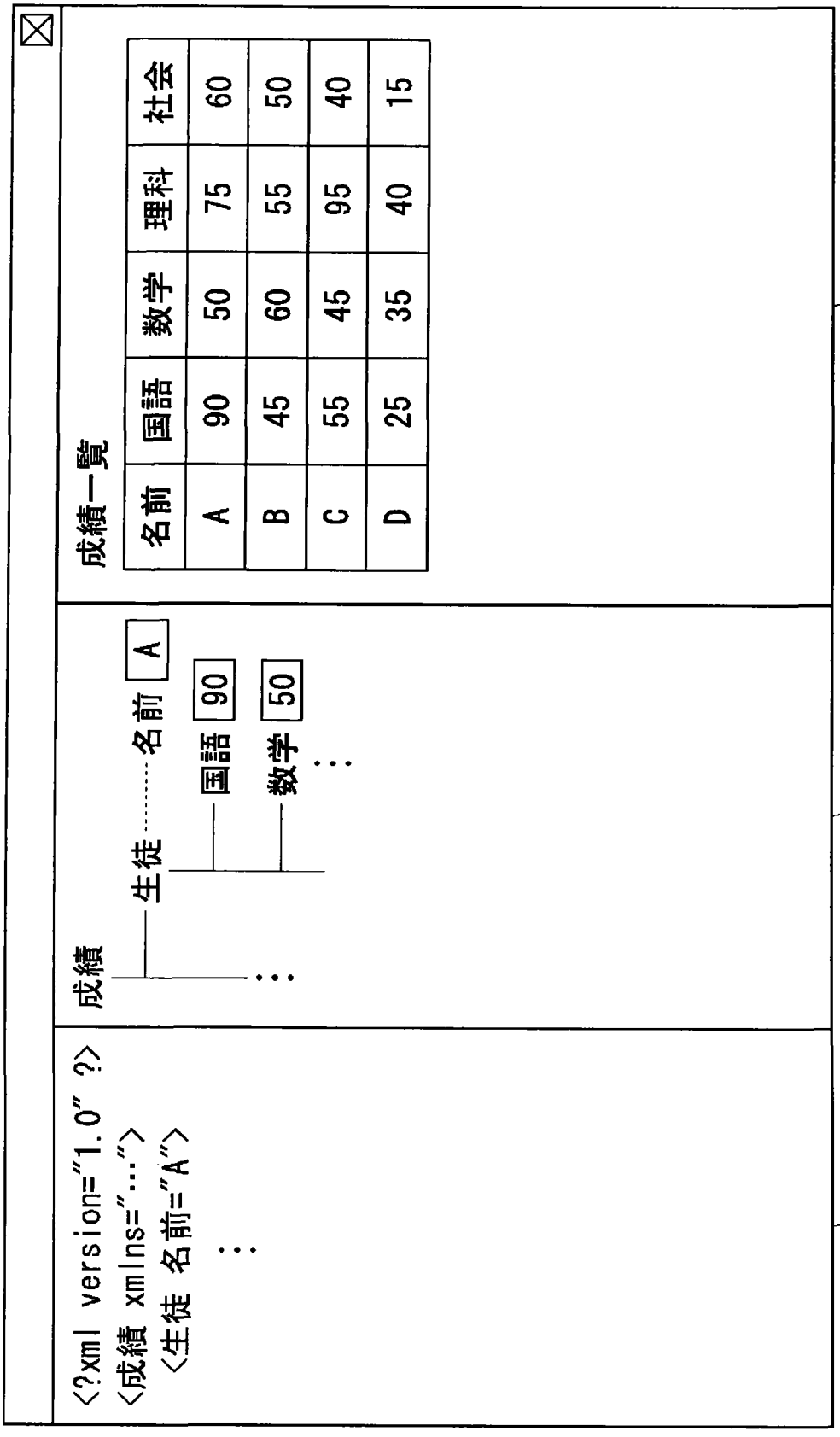

<?xml version="1.0" ?>
<成績 xmlns="...">
<生徒 名前="A">
...
成績
生徒
名前 A
国語 90
数学 50
...
成績一覧
名前 国語 数学 理科 社会
A 90 50 75 60
B 45 60 55 50
C 55 45 95 40
D 25 35 40 15
94
95
96

图 8

```
<?xml version="1.0" ?>

<svg xmlns="http://www.w3.org/2000/svg"
     width="400" height="200"
     viewBox="0 0 400 200"
>
  <rect x="-15" y="65" width="150" height="100" rx="20"
        transform="rotate(-20)"
        style="fill:none; stroke:purple; stroke-width:10"
  />
  <foreignObject x="190" y="10" width="200" height="200">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head><title /></head>
      <body bgcolor="#FFFFCC" text="darkgreen">
        <div style="font-size:12pt">
          SVG文書中に&lt;foreignObject&gt;を使って、
          XHTML文書を埋め込んでみました。
          数式も入れてみたりして:
          <div>
            <math xmlns="http://www.w3.org/1998/Math/MathML">
              <mi>x</mi>
              <mo>=</mo>
              <mfrac>
                <mrow>
                  <mo>−</mo>
                  <mi>b</mi>
                  <mo>±</mo>
                  <msqrt>
                    <mrow>
                      <msup>
                        <mi>b</mi>
                        <mn>2</mn>
                      </msup>
                      <mo>−</mo>
                      <mn>4</mn>
                      <mi>a</mi>
                      <mi>c</mi>
                    </mrow>
                  </msqrt>
                </mrow>
                <mrow>
                  <mn>2</mn>
                  <mi>a</mi>
                </mrow>
              </mfrac>
            </math>
          </div><!— math —>
        </div>
      </body>
    </html>
  </foreignObject>
</svg>
```

图 9

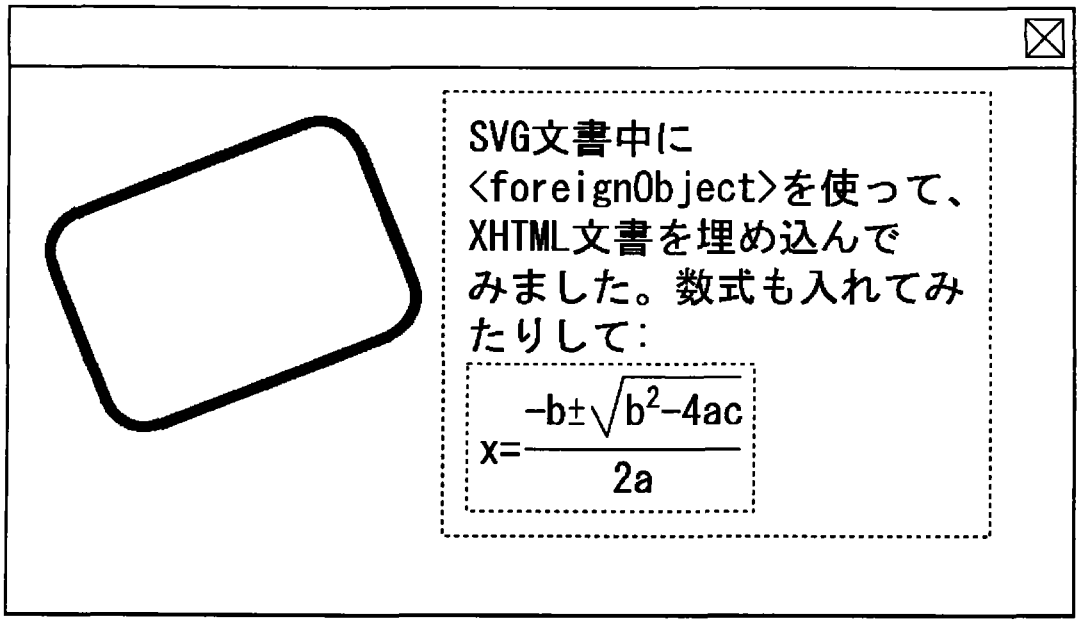

SVG文書中に
<foreignObject>を使って、
XHTML文書を埋め込んで
みました。数式も入れてみ
たりして:
$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$

图 10

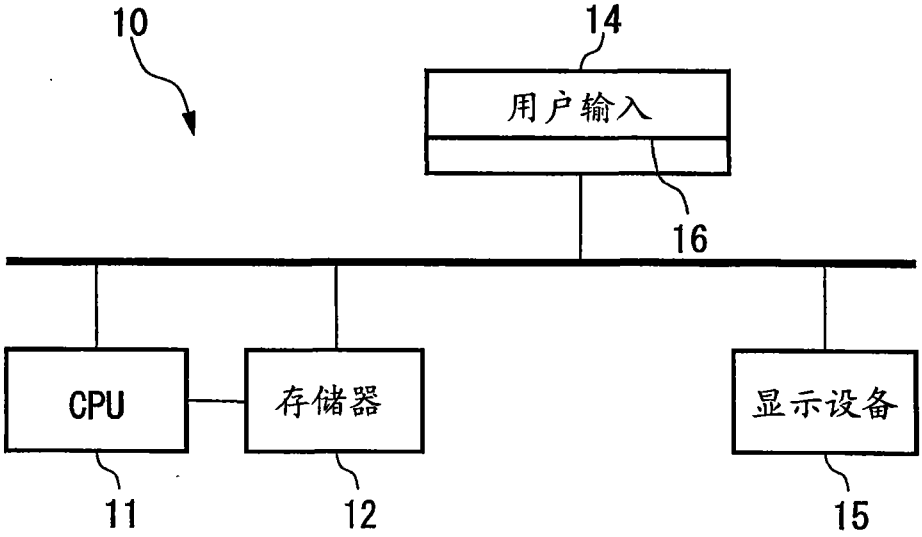

10
14
用户输入
16
CPU
存储器
显示设备
11
12
15

图 11(a)

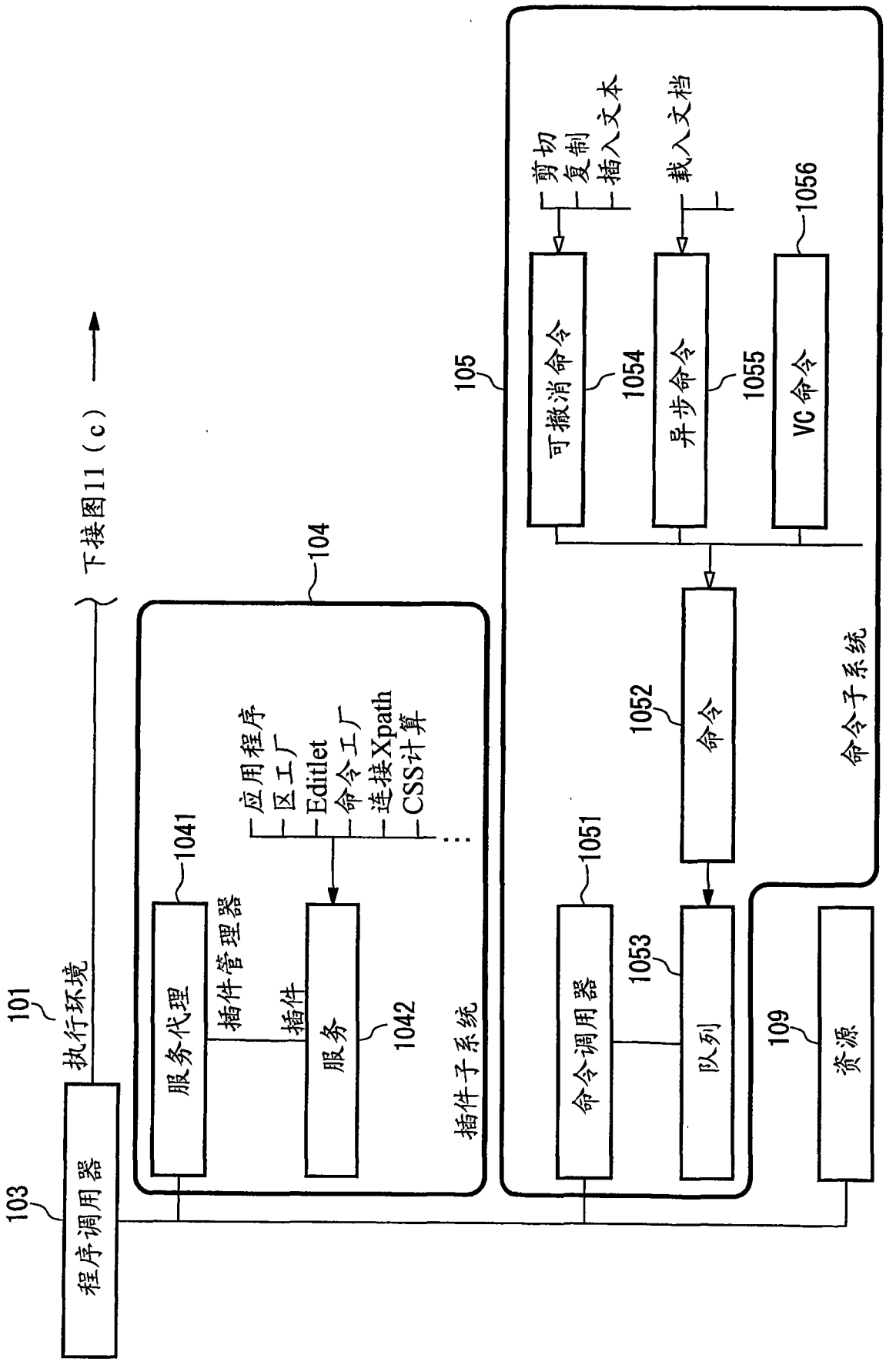

程序调用器
103
执行环境
101
下接图11（c）
插件子系统
104
服务代理
1041
插件管理器
插件
服务
1042
应用程序
区工厂
Editlet
命令工厂
连接Xpath
CSS计算
...
命令子系统
105
可撤消命令
1054
异步命令
1055
VC命令
1056
剪切
复制
插入文本
载入文档
命令
1052
命令调用器
1051
队列
1053
资源
109

图 11(b)

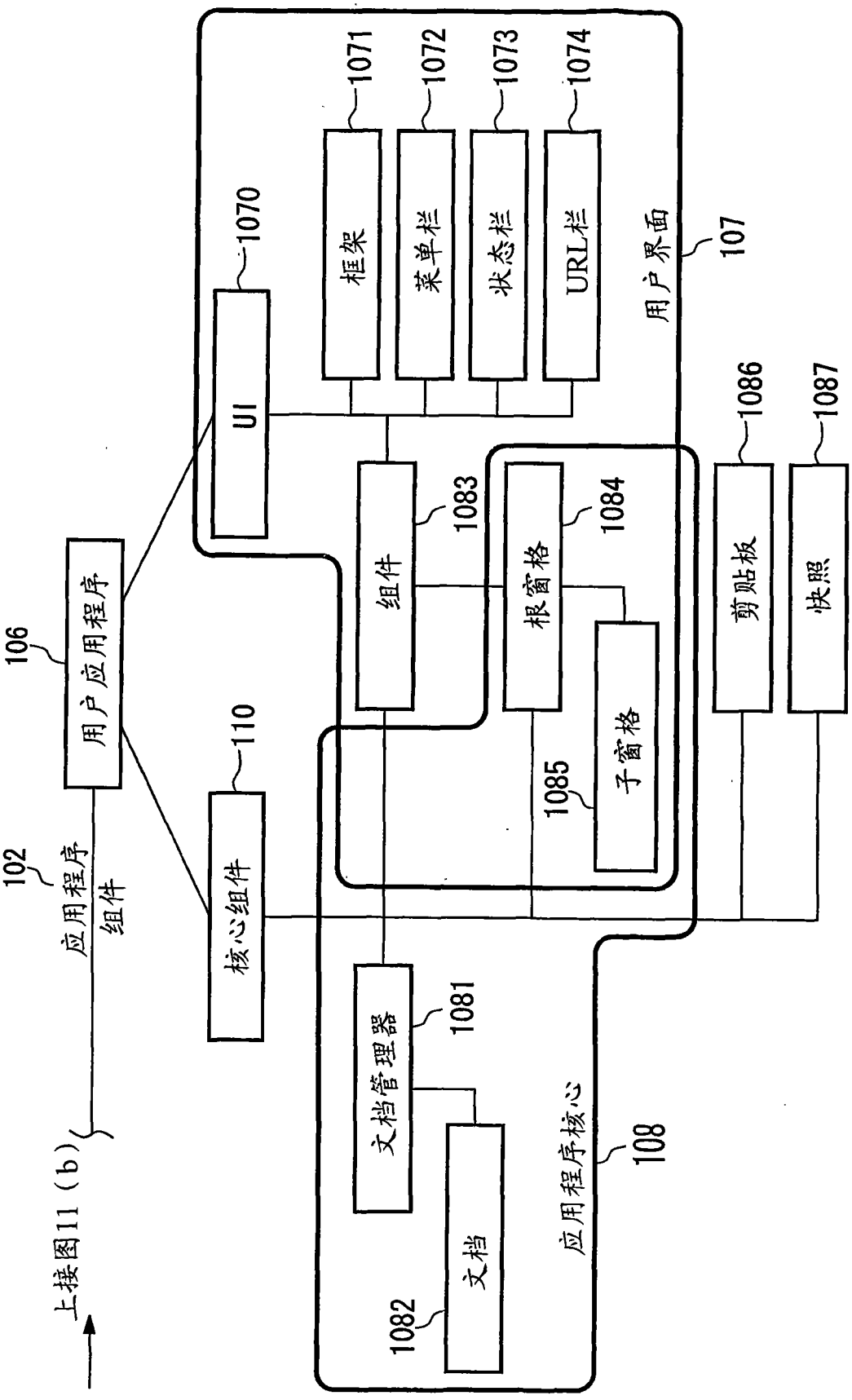

上接图11（b）
102
应用程序组件
106
用户应用程序
110
核心组件
UI
1070
框架
1071
菜单栏
1072
状态栏
1073
URL栏
1074
用户界面
107
组件
1083
根窗格
1084
子窗格
1085
剪贴板
1086
快照
1087
文档管理器
1081
文档
1082
应用程序核心
108

图 11(c)

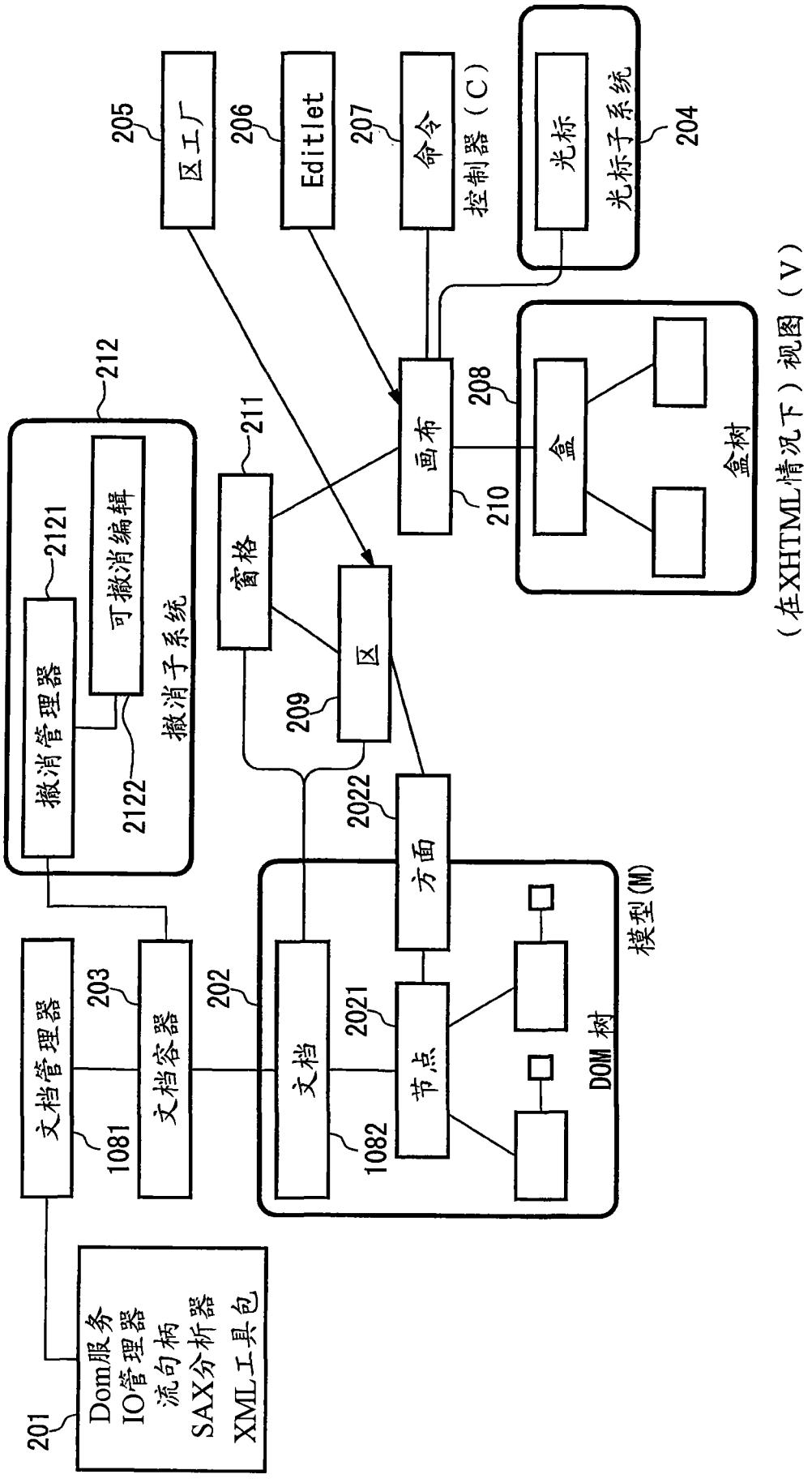

Dom服务
IO管理器
流句柄
SAX分析器
XML工具包
201
文档管理器
1081
203
文档容器
202
文档
1082
2021
节点
方面
2022
DOM 树
模型(M)
撤消管理器
2121
可撤消编辑
2122
撤消子系统
212
窗格
211
区
209
画布
208
210
盒
盒树
区工厂
205
Editlet
206
命令
207
控制器（C）
光标
光标子系统
204
（在XHTML情况下）视图（V）

图 12

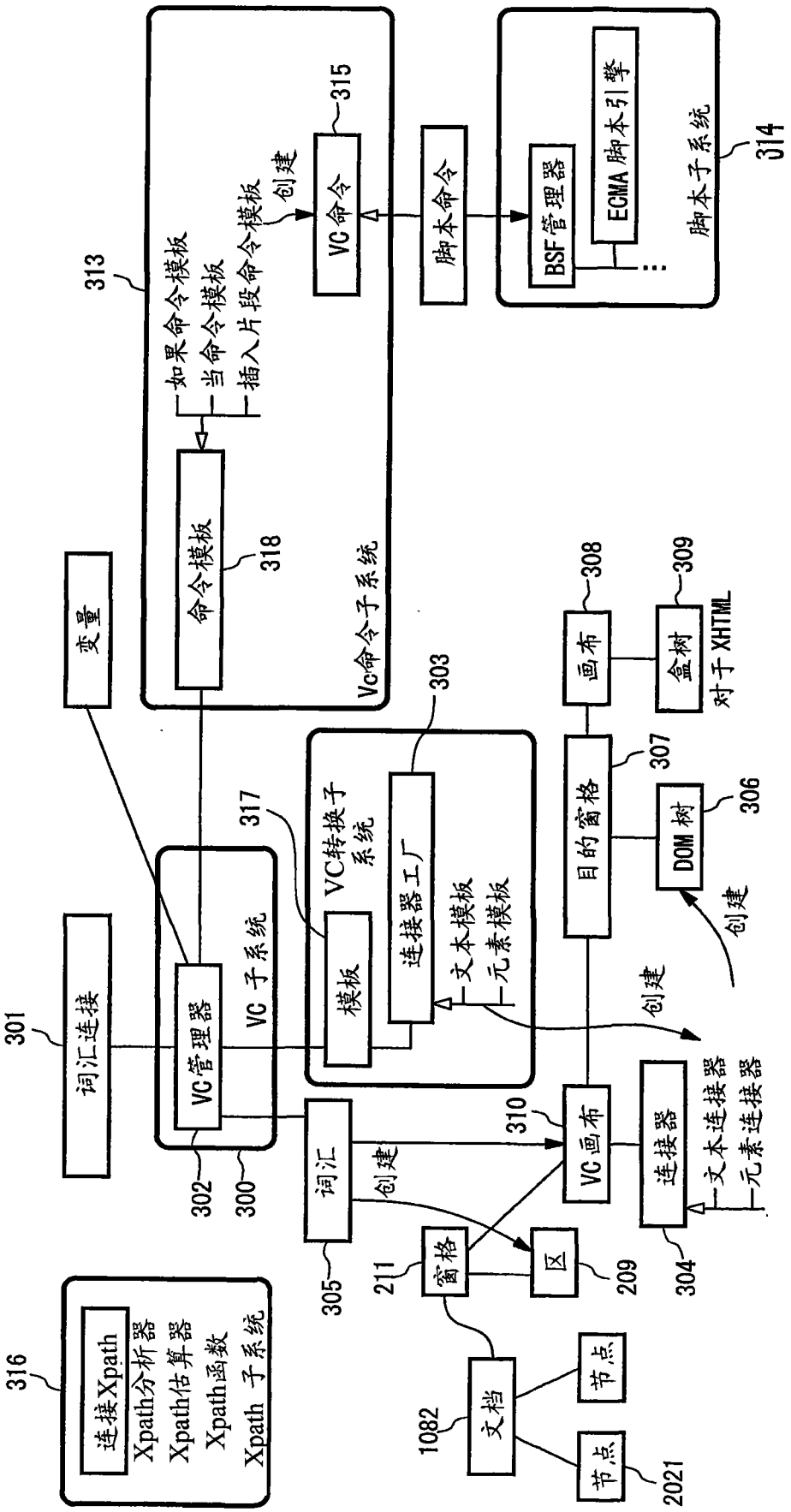

313
如果命令模板
当命令模板
插入片段命令模板
创建
VC 命令
315
脚本命令
BSF 管理器
ECMA 脚本引擎
脚本子系统
314
命令模板
318
Vc命令子系统
变量
301
词汇连接
VC管理器
VC 子系统
317
302
300
VC转换子系统
303
模板
连接器工厂
文本模板
元素模板
创建
画布
308
盒树
309
对于 XHTML
目的窗格
307
DOM 树
306
创建
词汇
305
310
VC 画布
连接器
文本连接器
元素连接器
211
窗格
区
209
304
316
连接Xpath
Xpath分析器
Xpath估算器
Xpath函数
Xpath 子系统
1082
文档
节点
节点
2021

图 13

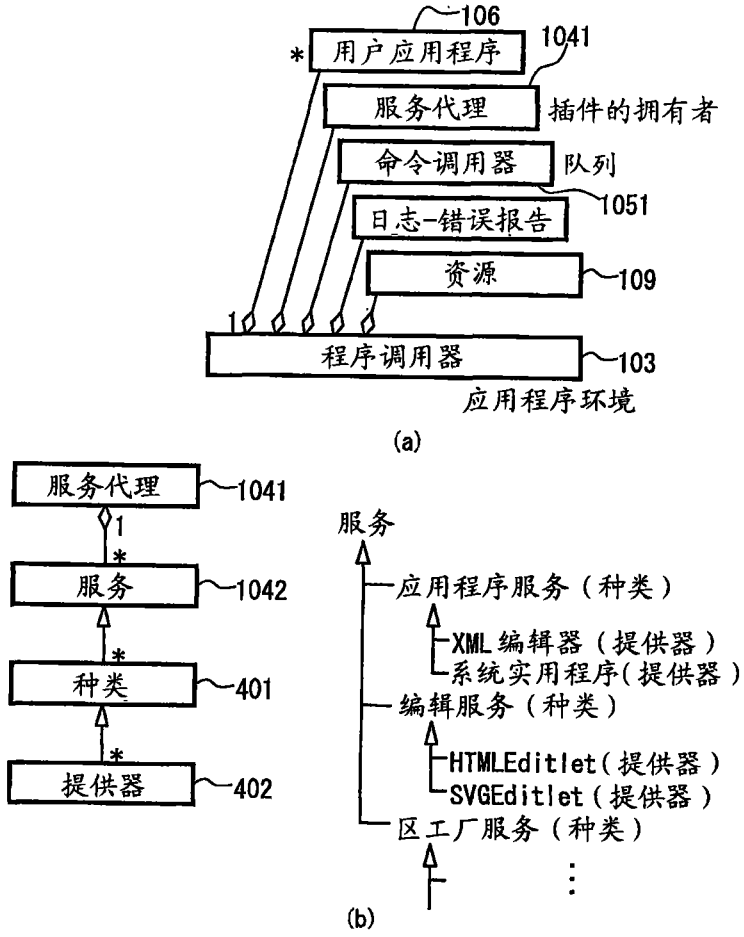

106
用户应用程序
1041
服务代理
插件的拥有者
命令调用器
队列
1051
日志-错误报告
资源
109
程序调用器
103
应用程序环境
(a)
服务代理
1041
服务
1042
种类
401
提供器
402
(b)
服务
应用程序服务（种类）
XML编辑器（提供器）
系统实用程序（提供器）
编辑服务（种类）
HTMLEditlet（提供器）
SVGEditlet（提供器）
区工厂服务（种类）
(c)
服务代理
1041
种类
提供器
402
401
(d)
程序调用器
用户应用程序
103
1041
106
插件服务器
服务调用器
创建
载入
插件
存储
(e)

图 14

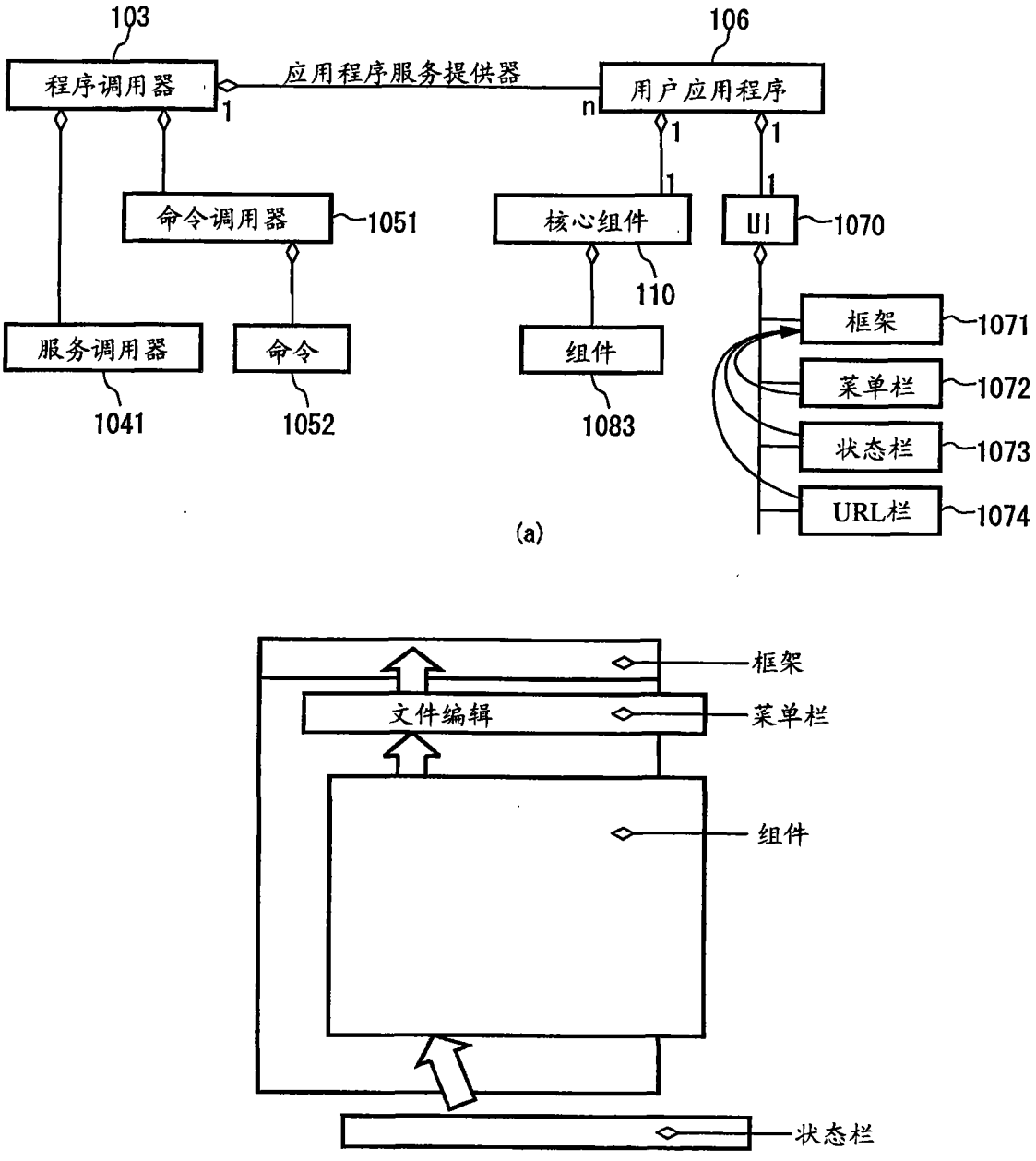
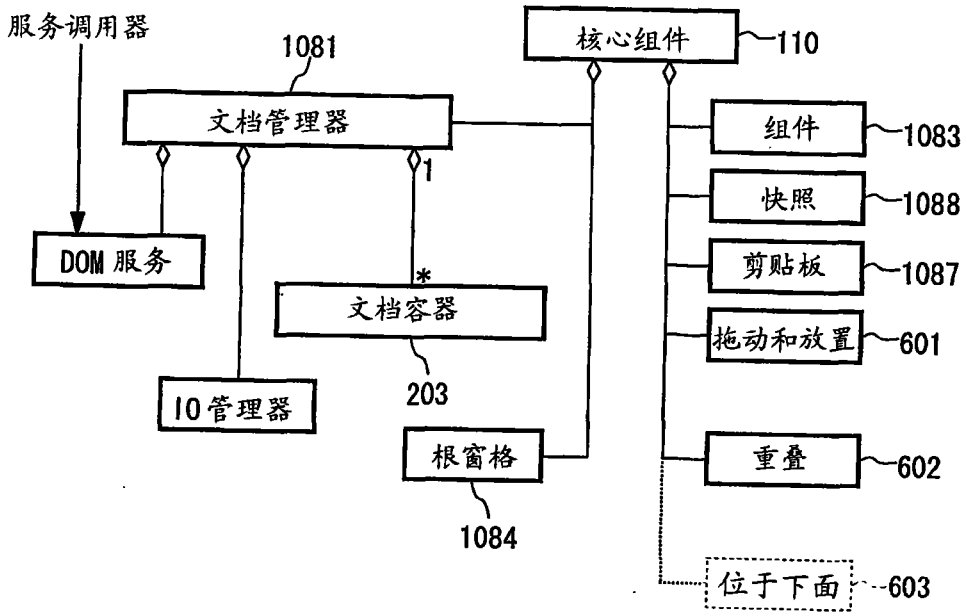
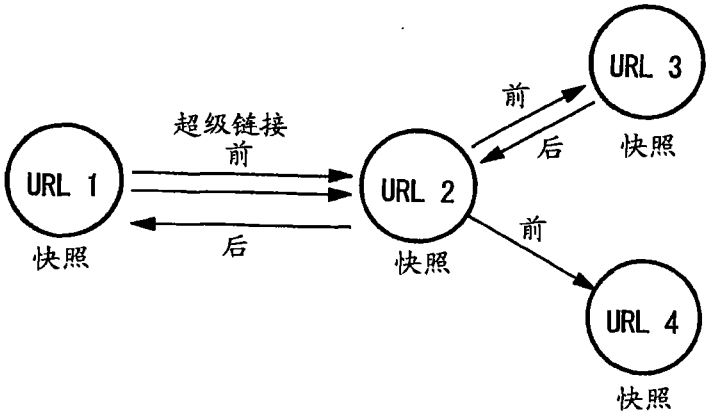

103
程序调用器
1
应用程序服务提供器
n
106
用户应用程序
1
1
1
1
命令调用器
1051
核心组件
UI
1070
110
框架
1071
菜单栏
1072
状态栏
1073
URL栏
1074
服务调用器
命令
组件
1041
1052
1083
(a)
框架
文件编辑
菜单栏
组件
状态栏
(b)

图 15

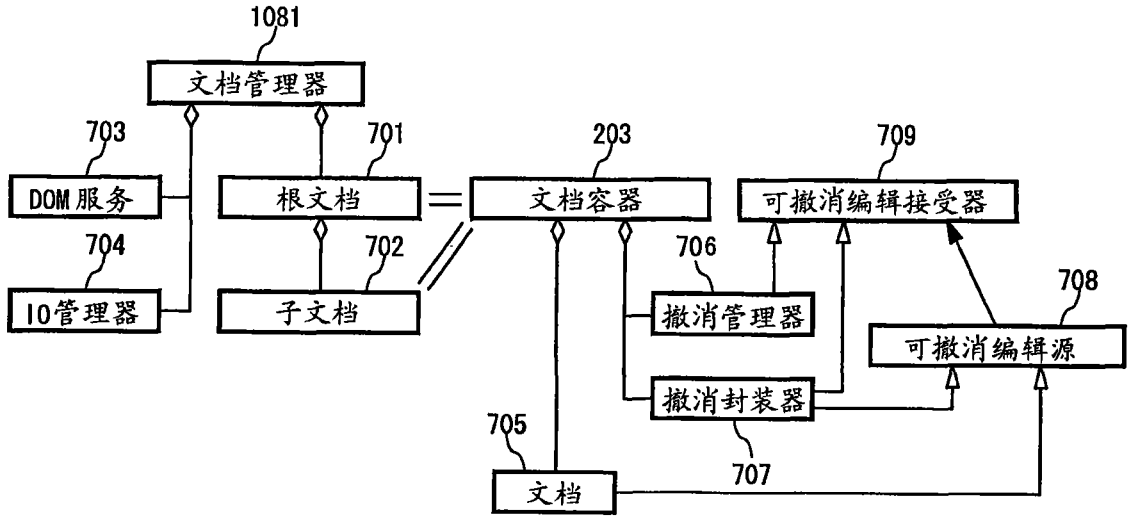

服务调用器
1081
核心组件
110
文档管理器
组件
1083
1
快照
1088
剪贴板
1087
DOM 服务
*
文档容器
拖动和放置
601
203
IO管理器
根窗格
重叠
602
1084
位于下面
603

(a)

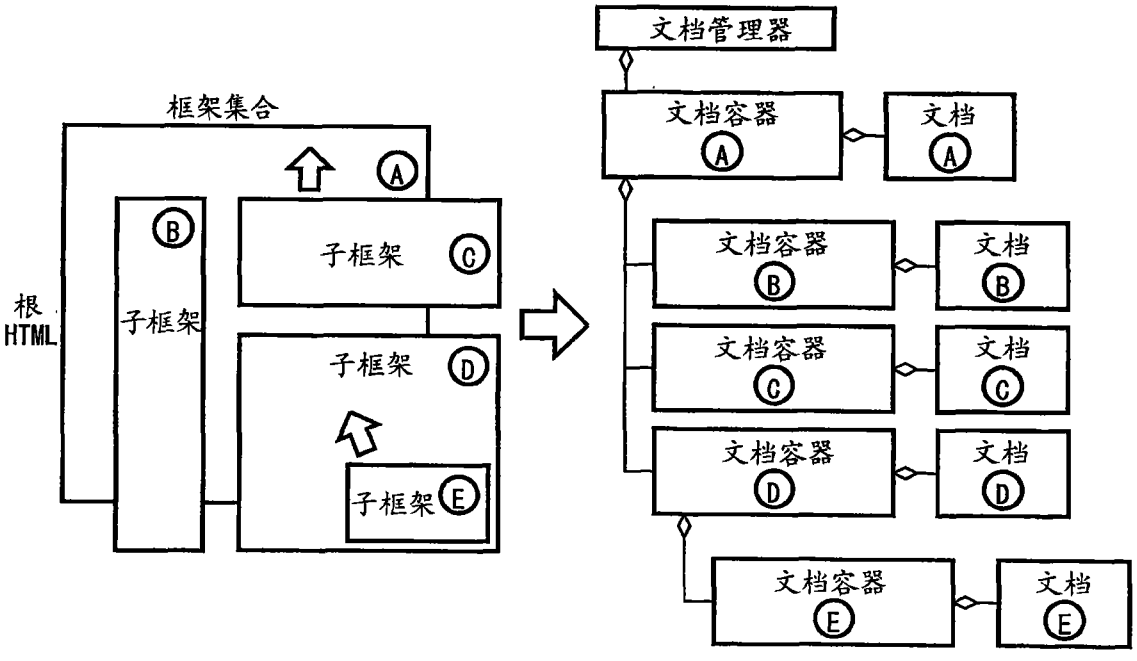

URL 3
前
超级链接
前
后
快照
URL 1
URL 2
前
后
快照
快照
URL 4
快照

(b)

图 16


1081
文档管理器
703
DOM服务
704
IO管理器
701
根文档
702
子文档
203
文档容器
709
可撤消编辑接受器
706
撤消管理器
708
可撤消编辑源
撤消封装器
705
文档
707

(a)


框架集合
根
HTML
A
B
子框架
C
子框架
D
子框架
E
文档管理器
文档容器
A
文档
A
文档容器
B
文档
B
文档容器
C
文档
C
文档容器
D
文档
D
文档容器
E
文档
E

(b)

图 17

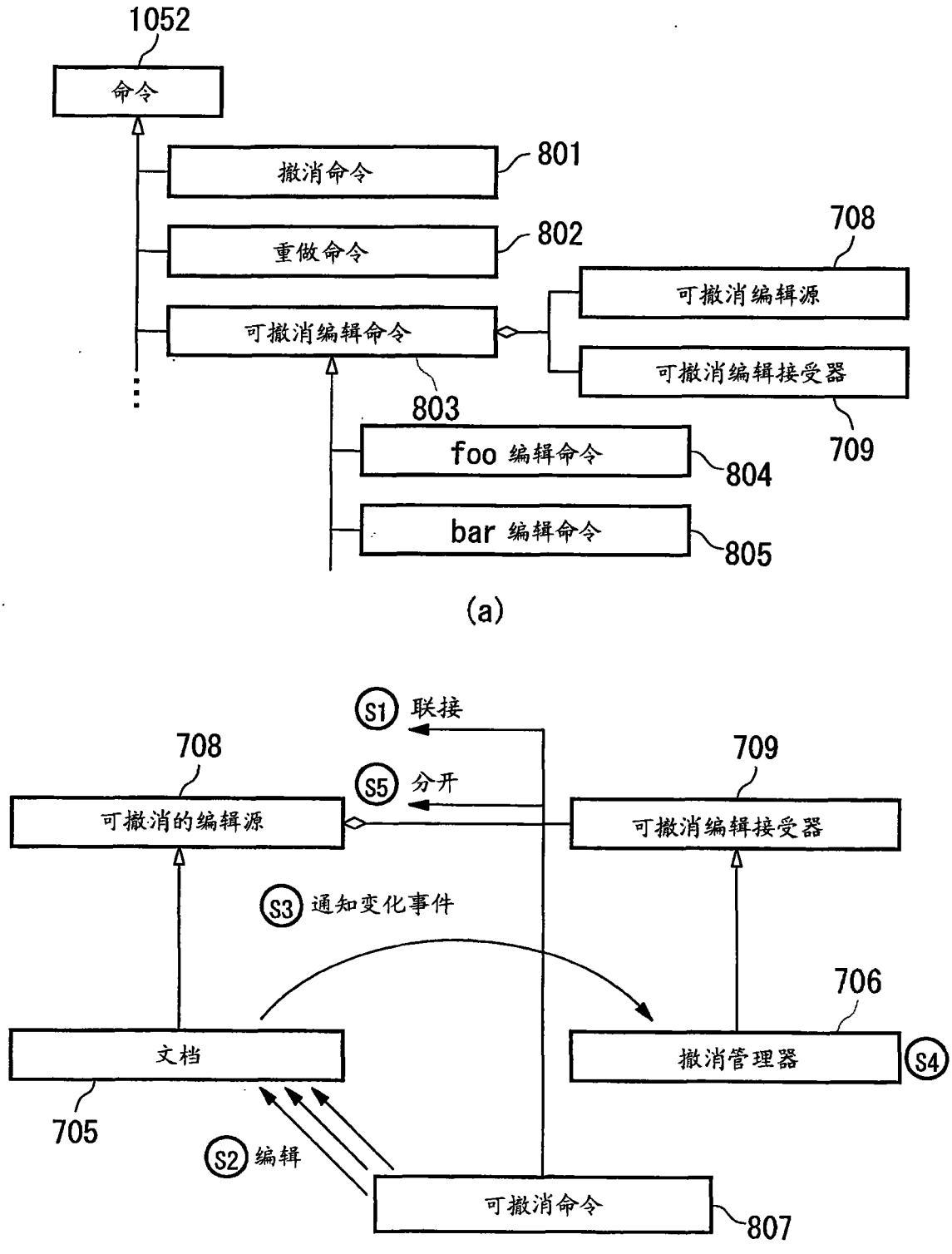
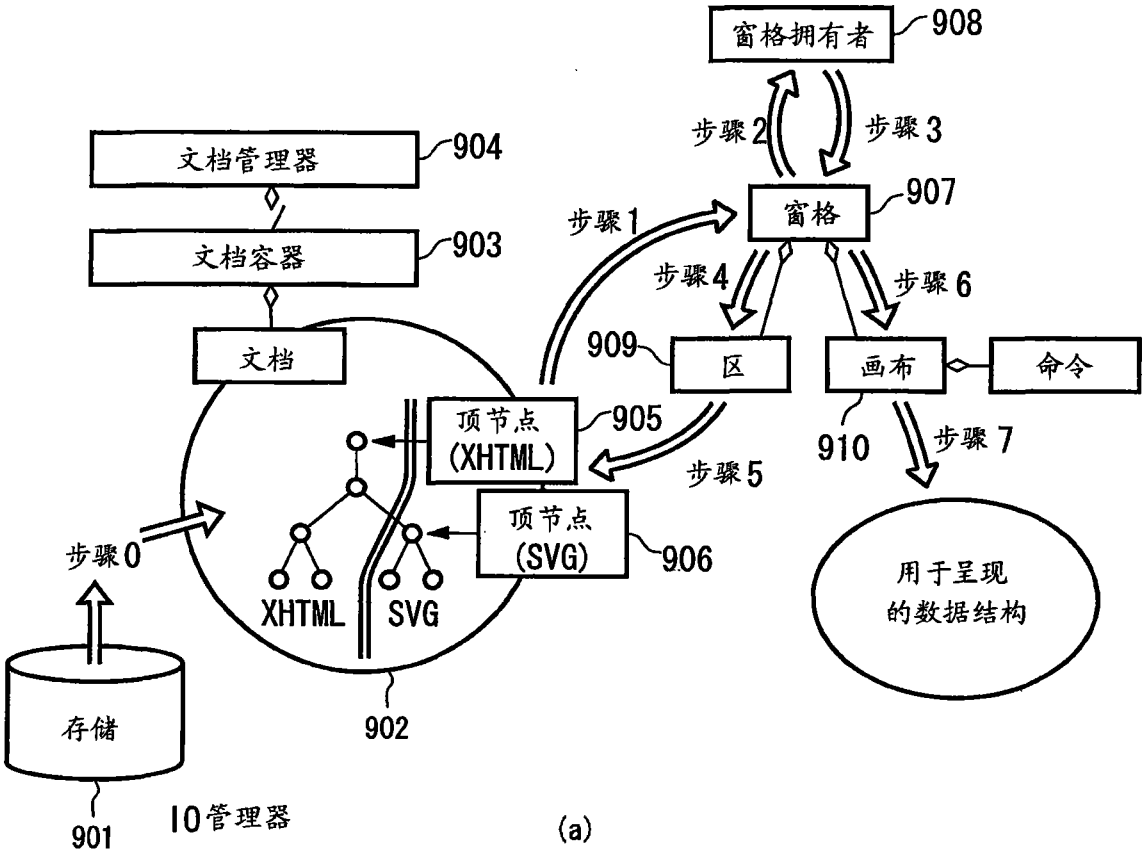

1052
命令
撤消命令
801
重做命令
802
708
可撤消编辑源
可撤消编辑命令
可撤消编辑接受器
803
709
foo 编辑命令
804
bar 编辑命令
805
(a)
S1 联接
708
709
S5 分开
可撤消的编辑源
可撤消编辑接受器
S3 通知变化事件
706
文档
撤消管理器
S4
705
S2 编辑
可撤消命令
807
(b)

图 18


窗格拥有者
908
步骤2
步骤3
文档管理器
904
窗格
907
步骤1
文档容器
903
步骤4
步骤6
文档
909
区
画布
命令
顶节点
(XHTML)
905
910
步骤7
步骤5
步骤0
顶节点
(SVG)
906
用于呈现
的数据结构
XHTML
SVG
存储
902
901
IO管理器
(a)

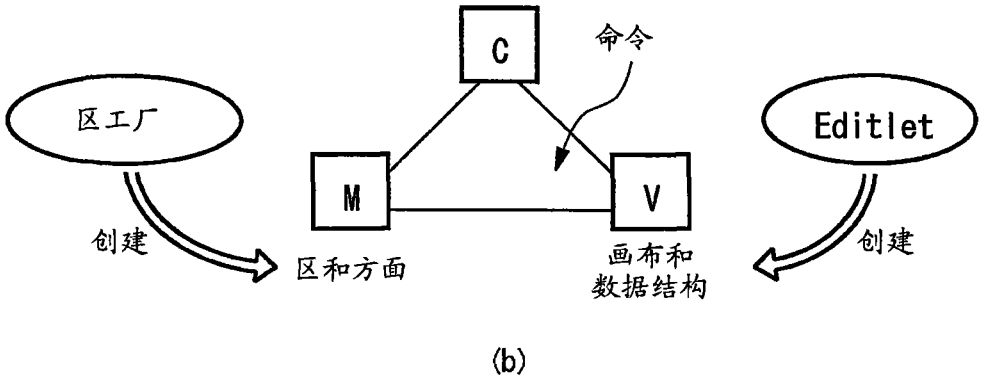

C
命令
区工厂
Editlet
M
V
创建
区和方面
画布和
数据结构
创建
(b)

图 19

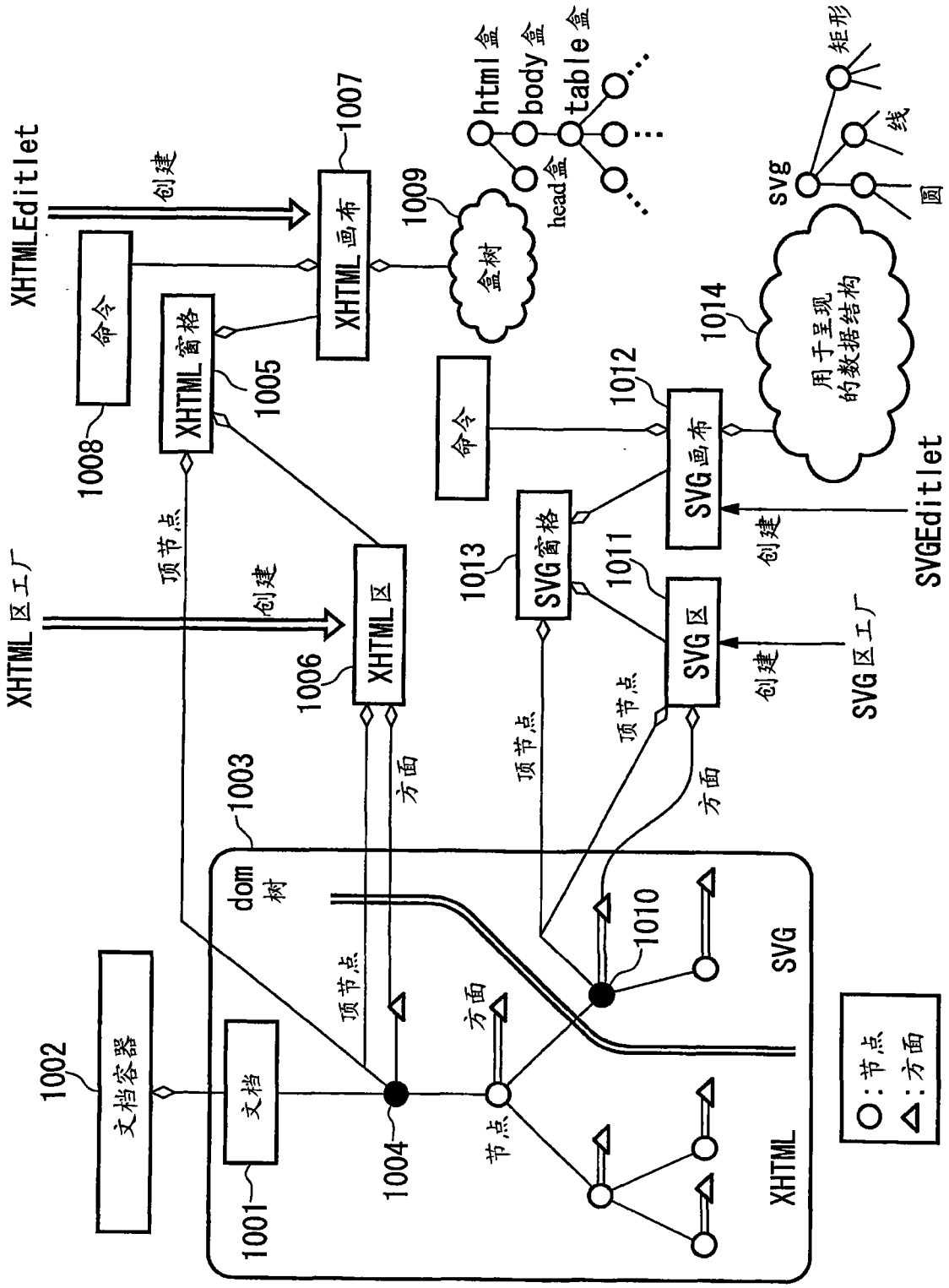

XHTMLEditlet
创建
命令
1008
XHTML窗格
1005
XHTML画布
1007
1009
盒树
html盒
body盒
table盒
head盒
XHTML区工厂
顶节点
创建
XHTML区
1006
1003
dom树
方面
1002
文档容器
文档
1001
1004
节点
顶节点
方面
1010
SVG
XHTML
命令
1012
SVG画布
1014
用于呈现的数据结构
SVG
矩形
线
圆
SVGEditlet
1013
SVG窗格
1011
SVG区
创建
SVG区工厂
顶节点
方面
○:节点
△:方面

图 20

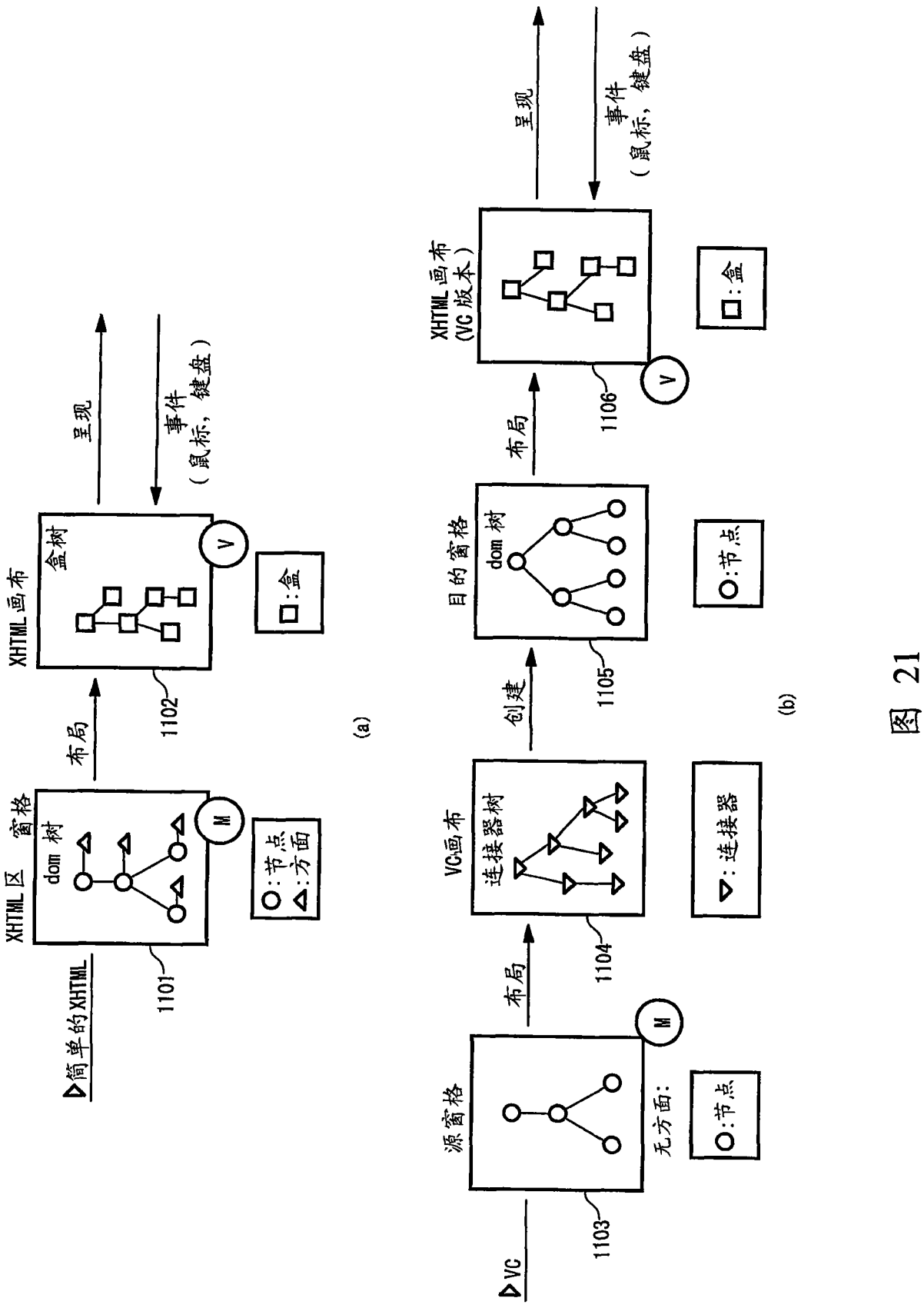
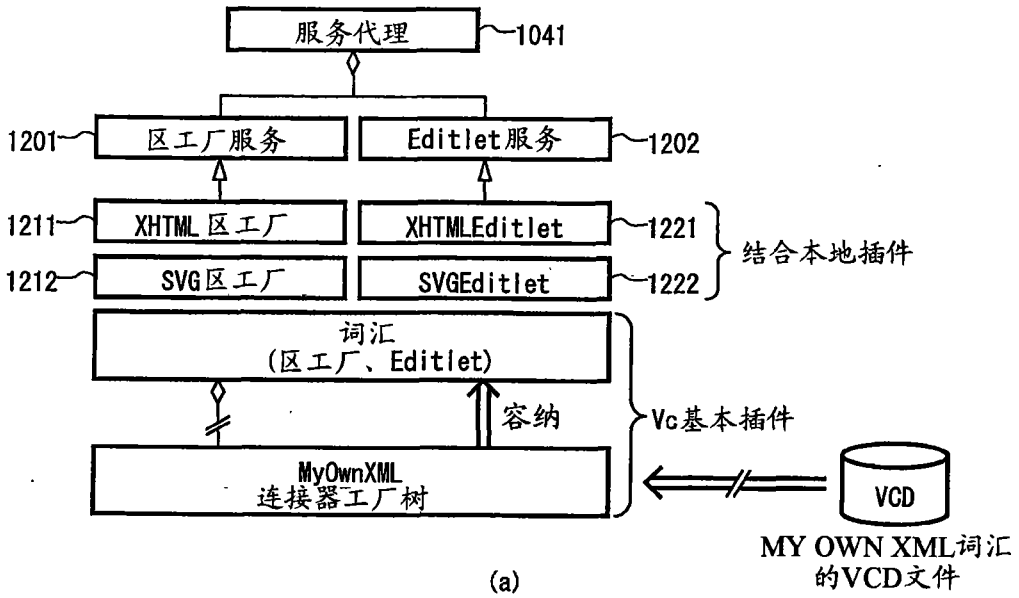
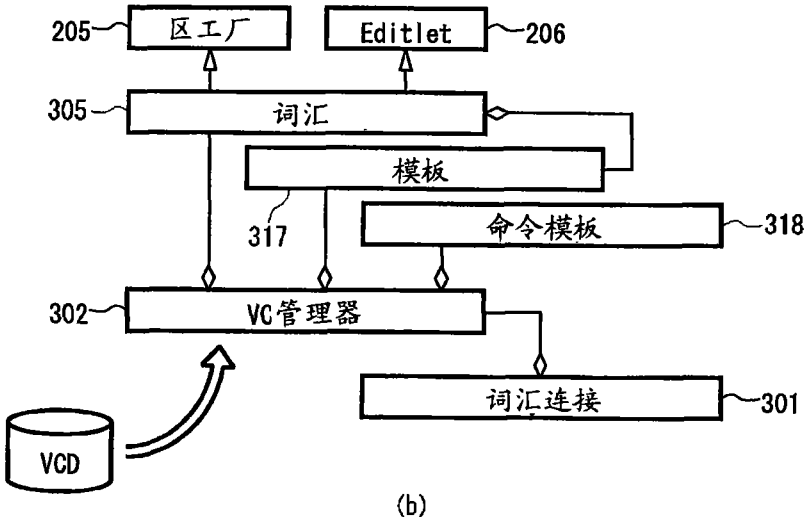

简单的XHTML
XHTML 区 窗格
dom 树
M
1101
布局
XHTML 画布
盒树
V
1102
呈现
事件
（鼠标，键盘）
O:节点
△:方面
□:盒
(a)
VC
源窗格
M
无方面:
O:节点
1103
布局
VC画布
连接器树
1104
创建
目的窗格
dom 树
1105
布局
XHTML 画布
(VC 版本)
V
1106
呈现
事件
（鼠标，键盘）
▽:连接器
O:节点
□:盒
(b)

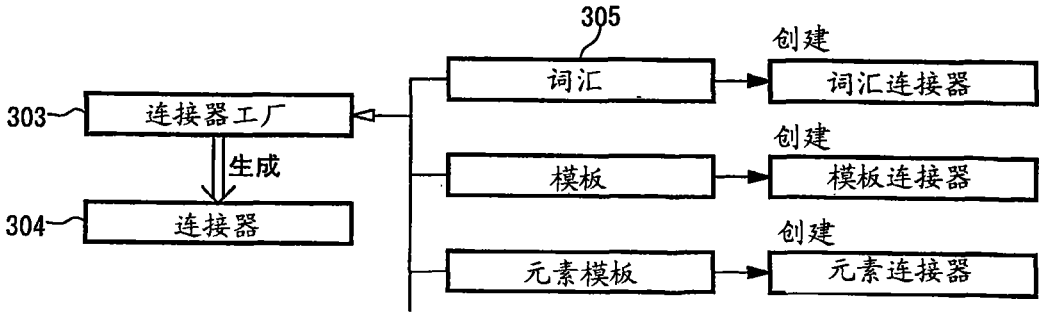
图 21


服务代理
1041
1201
区工厂服务
Editlet服务
1202
1211
XHTML区工厂
XHTMLEditlet
1221
结合本地插件
1212
SVG区工厂
SVGEditlet
1222
词汇
(区工厂、Editlet)
容纳
Vc基本插件
MyOwnXML
连接器工厂树
VCD
MY OWN XML词汇
的VCD文件

(a)


205
区工厂
Editlet
206
305
词汇
模板
317
命令模板
318
302
VC管理器
词汇连接
301
VCD

(b)


305
创建
词汇
词汇连接器
303
连接器工厂
创建
生成
模板
模板连接器
304
连接器
创建
元素模板
元素连接器

(c)

图 22

用于MYSAMPLE XML的VCD文件

```
<?xml version="1.0"?>
<vcd:vcd version="0.1"
         xmlns="http://www.w3.org/1999/xhtml"
         xmlns:vcd="http://xmlns.xfytec.com/vcd"
         xmlns:function="http://xmlns.xfytec.com/function"
         xmlns:sample="http://www.justsystem.com/sample">
<vcd:vocabulary match="sample:root"
                label="My Sample XML"
                call-template="SampleTemplate"/>
<vcd:template name="SampleTemplate">
  <html>
              <head>
      <title>
        <vcd:value-of select="function:file-name()"/>
      </title>
              </head>
              <body>
      <p>
        <vcd:text-of select="." type="vcd:plain-text"/>
      </p>
              </body>
  </html>
</vcd:template>
</vcd:vcd>
```

用于MYSAMPLE XML的Vc管理器

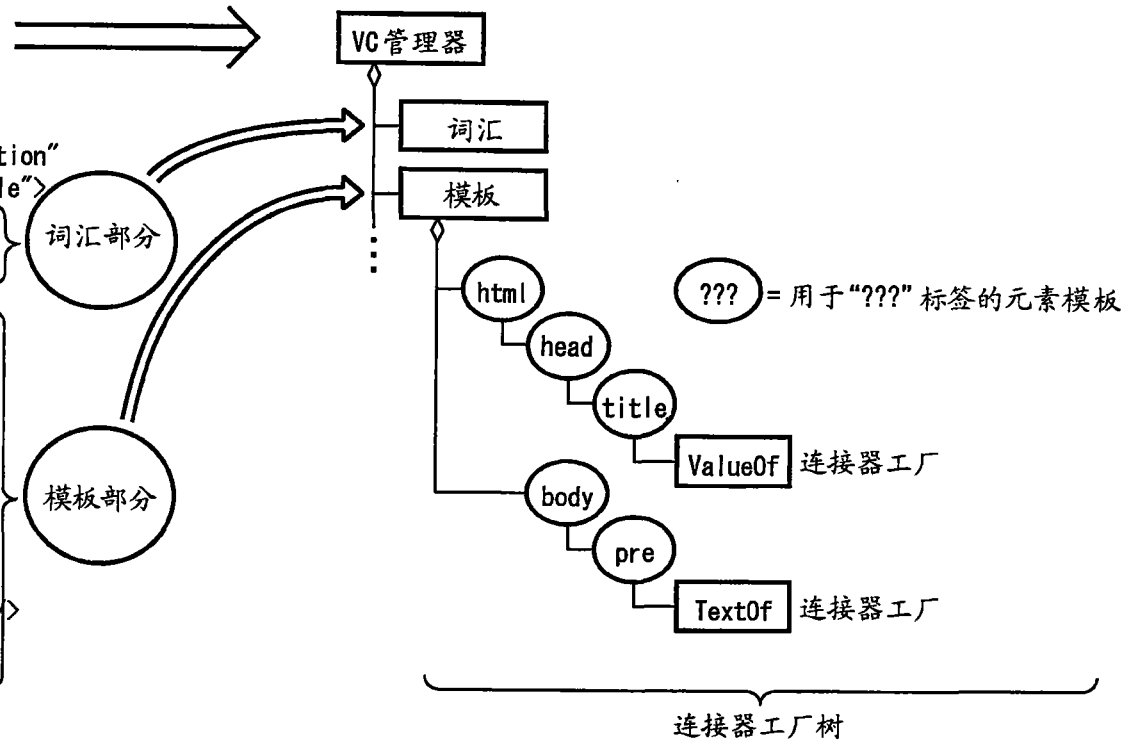
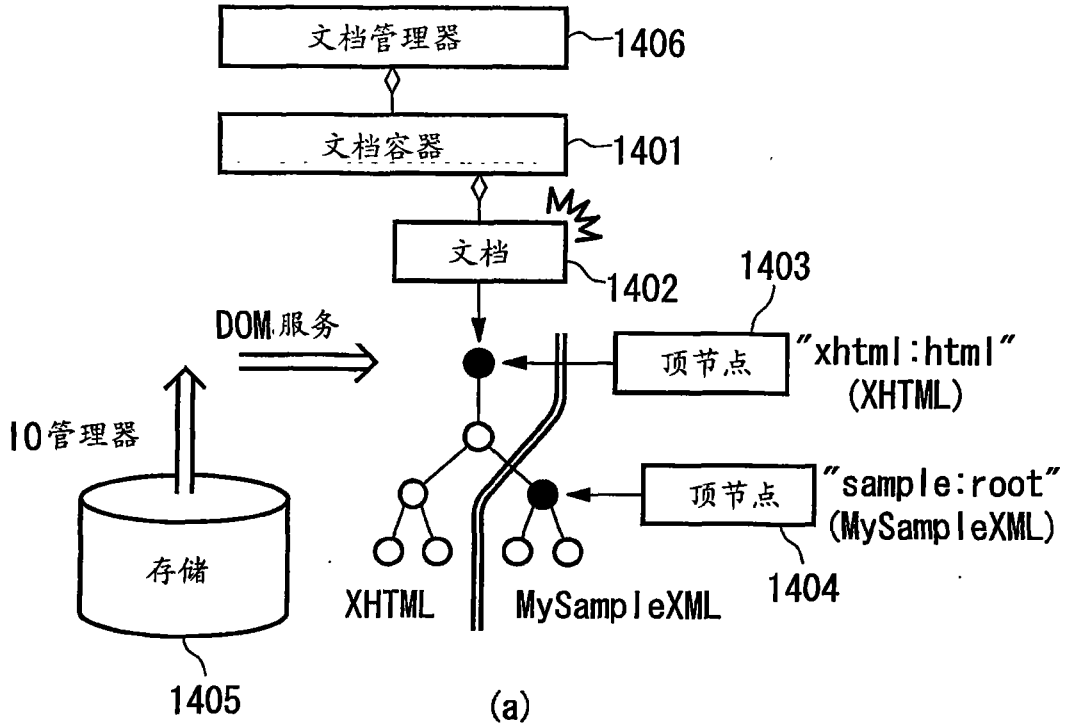
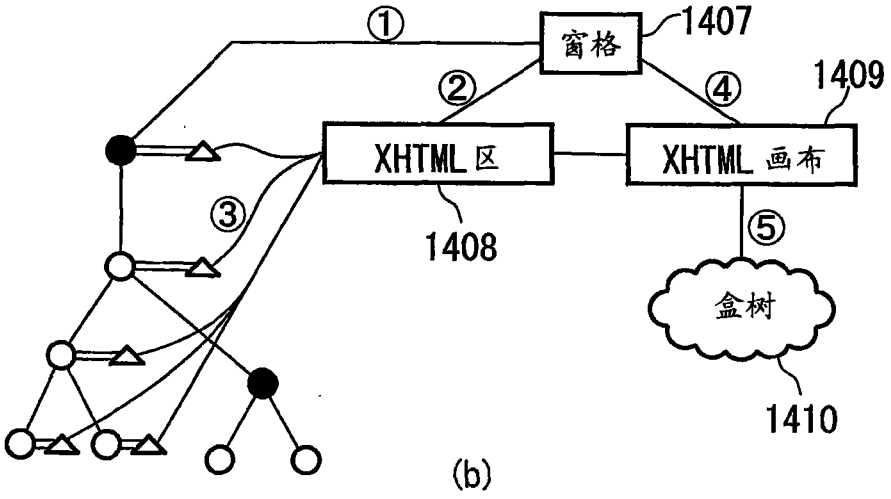


图 23


文档管理器
1406
文档容器
1401
文档
1402
1403
DOM服务
顶节点
"xhtml:html"
(XHTML)
IO管理器
存储
顶节点
"sample:root"
(MySampleXML)
1404
XHTML
MySampleXML
1405
(a)


①
窗格
1407
②
④
1409
XHTML区
XHTML画布
③
⑤
1408
盒树
1410
(b)

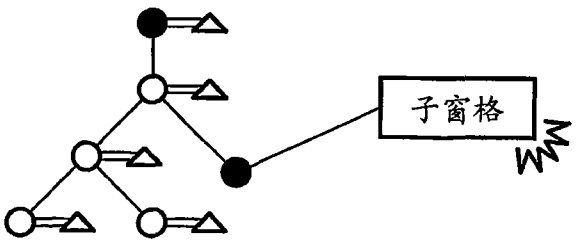

子窗格
(c)

图 24

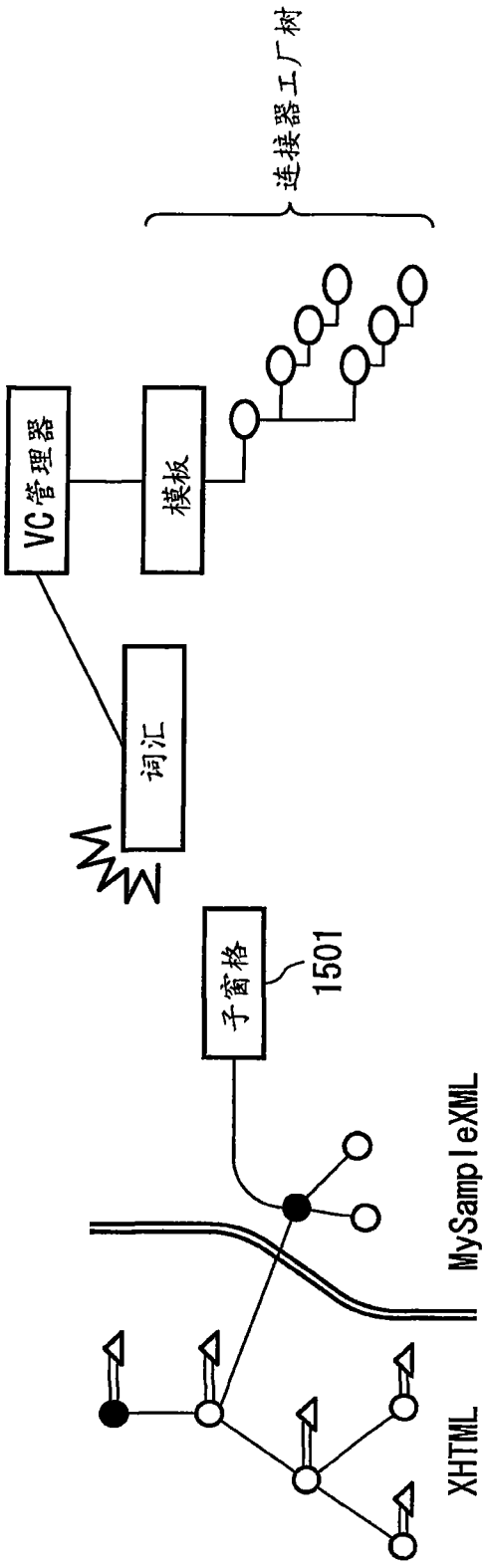


图 25

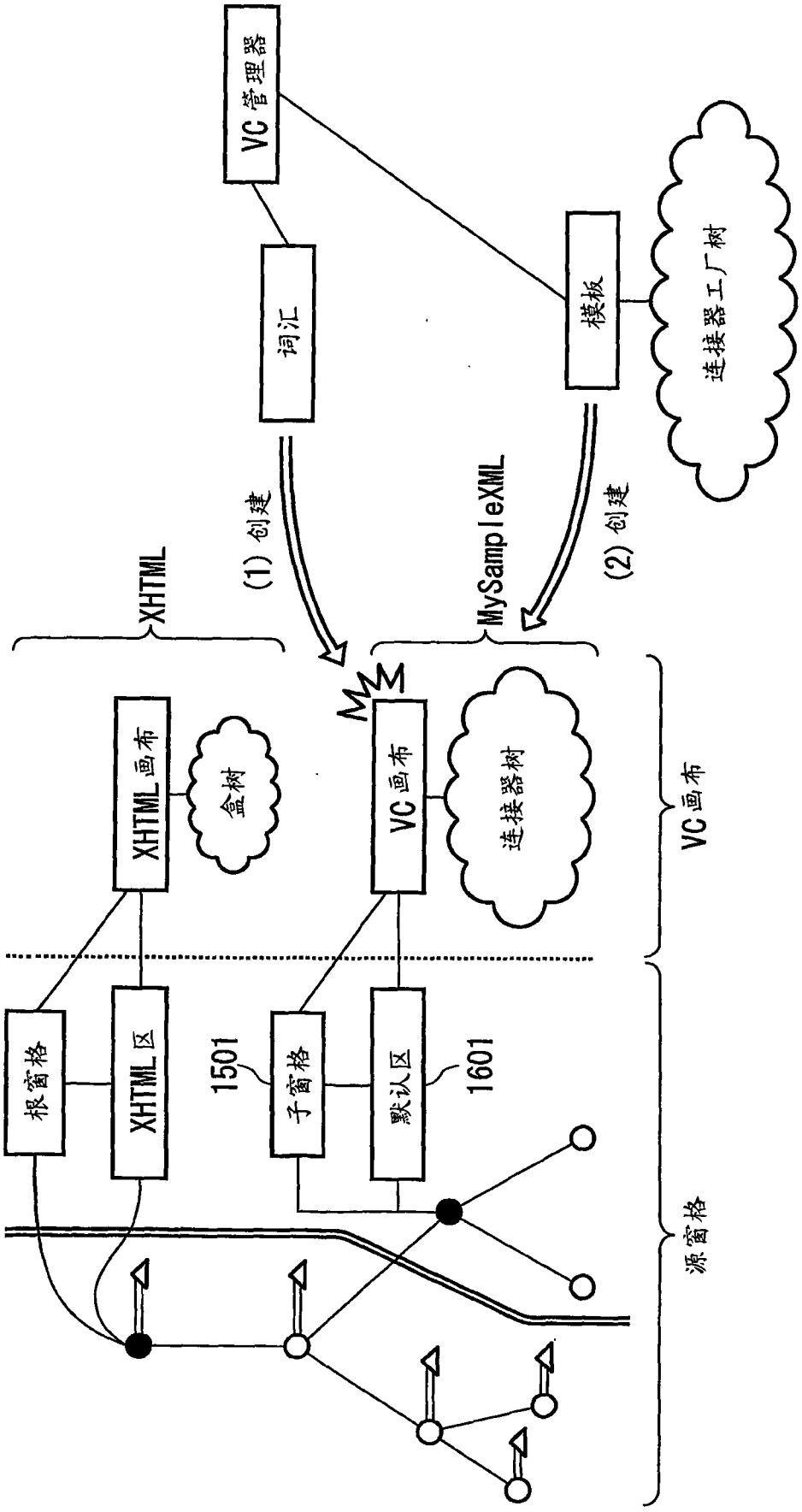

VC 管理器
词汇
模板
连接器工厂树
(1) 创建
(2) 创建
XHTML
MySampleXML
XHTML 画布
盒树
VC 画布
连接器树
根窗格
XHTML 区
1501
子窗格
默认区
1601
VC 画布
源窗格

图 26

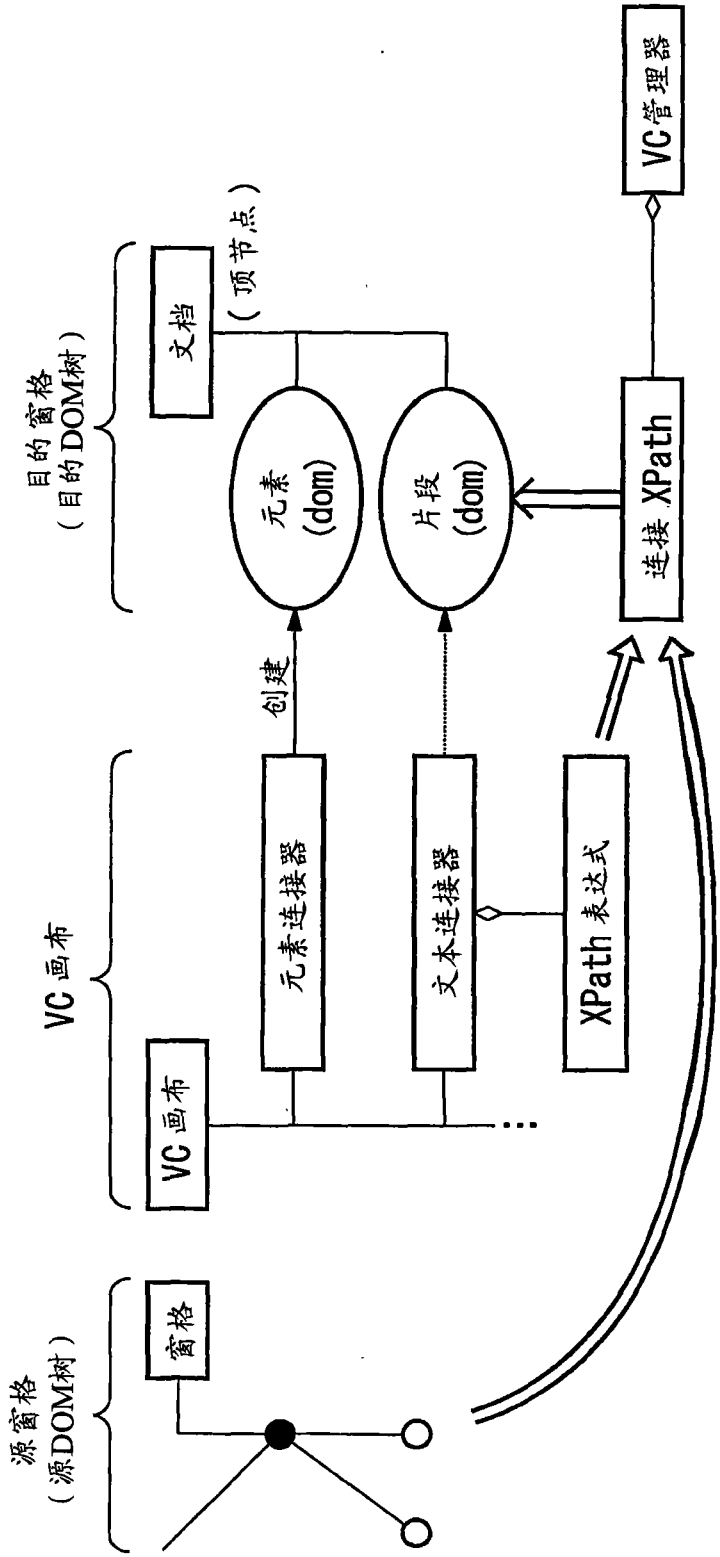

源窗格
（源DOM树）
窗格
VC 画布
VC 画布
元素连接器
文本连接器
XPath 表达式
创建
目的窗格
（目的DOM树）
文档
（顶节点）
元素
(dom)
片段
(dom)
连接 XPath
VC管理器

图 27

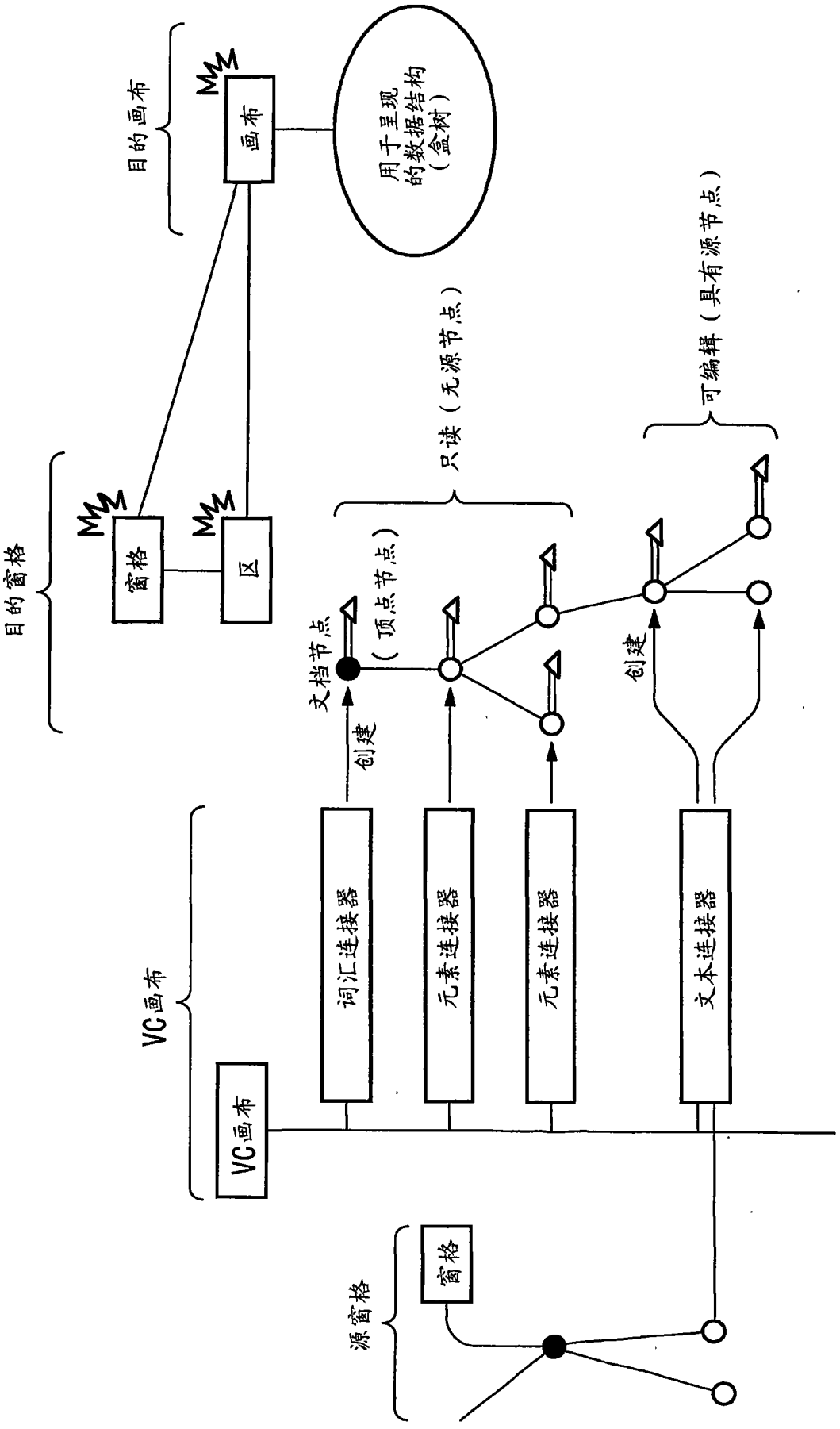

目的画布
画布
用于呈现的数据结构（盒树）
目的窗格
窗格
区
只读（无源节点）
可编辑（具有源节点）
文档节点（顶点节点）
创建
创建
VC画布
VC画布
词汇连接器
元素连接器
元素连接器
文本连接器
源窗格
窗格

图 28

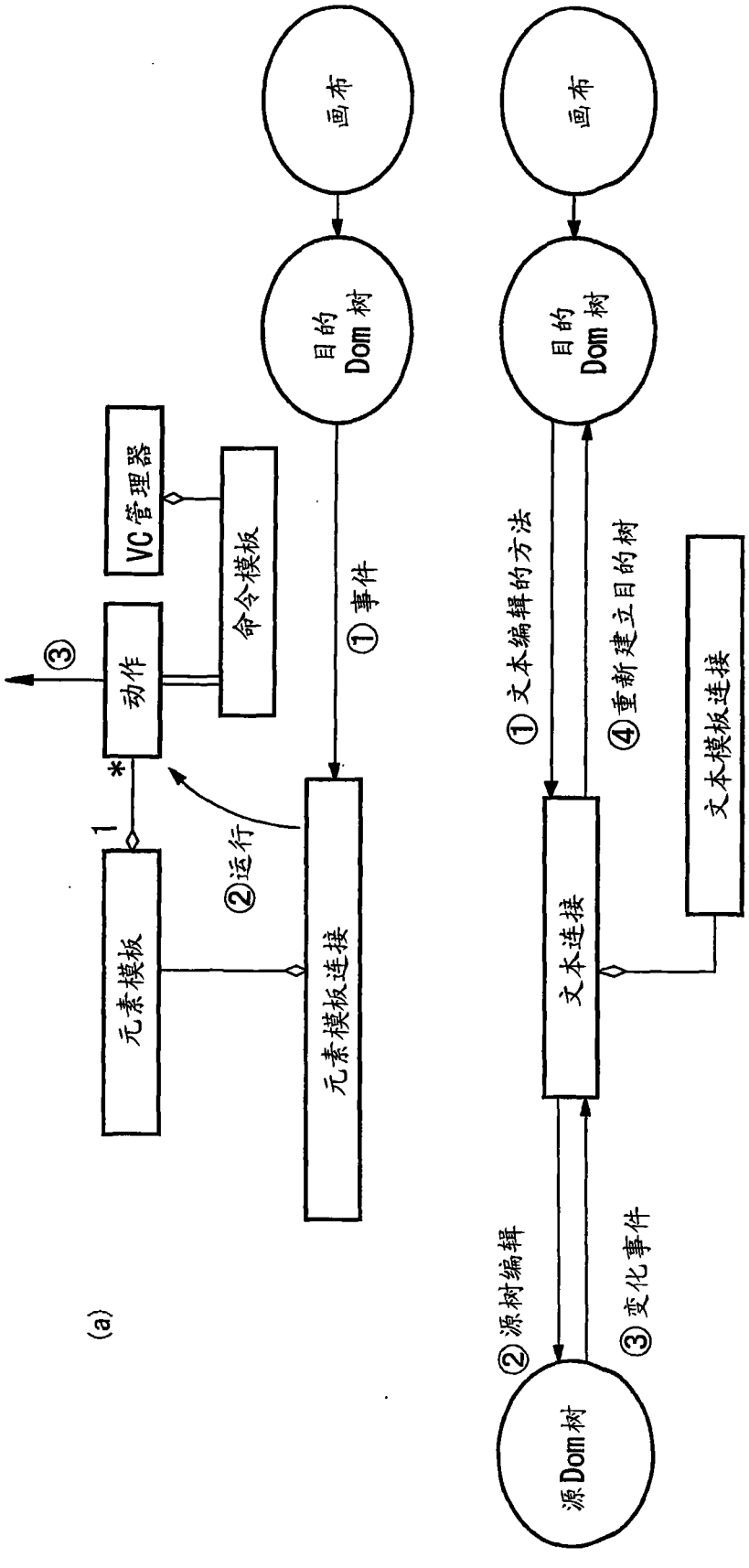

(a)
画布
目的 Dom 树
① 事件
VC管理器
命令模板
③
动作
*
1
元素模板
②运行
元素模板连接
(b)
画布
目的 Dom 树
① 文本编辑的方法
④ 重新建立目的树
文本连接
文本模板连接
② 源树编辑
③ 变化事件
源Dom树

图 29

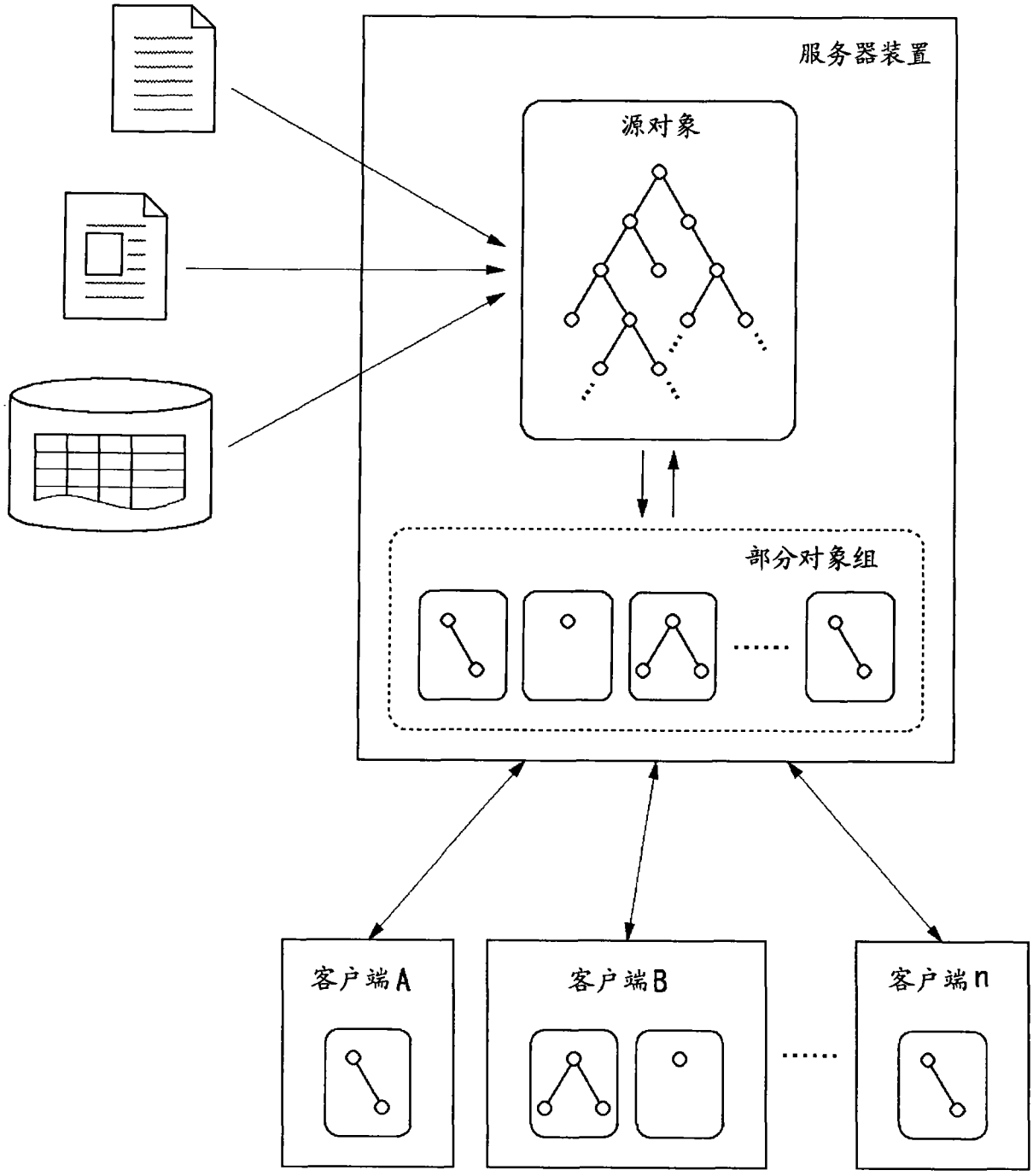


图 30

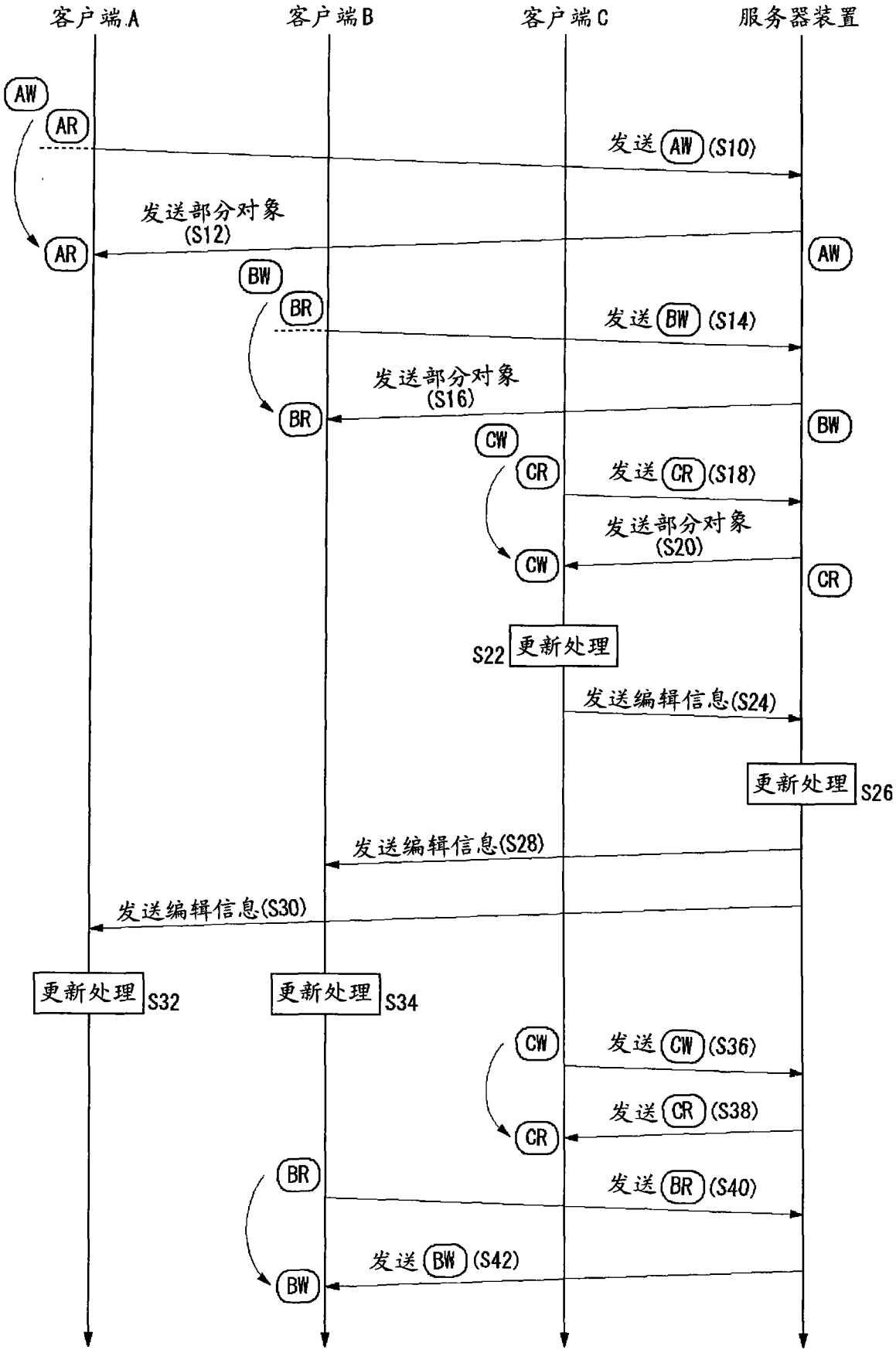
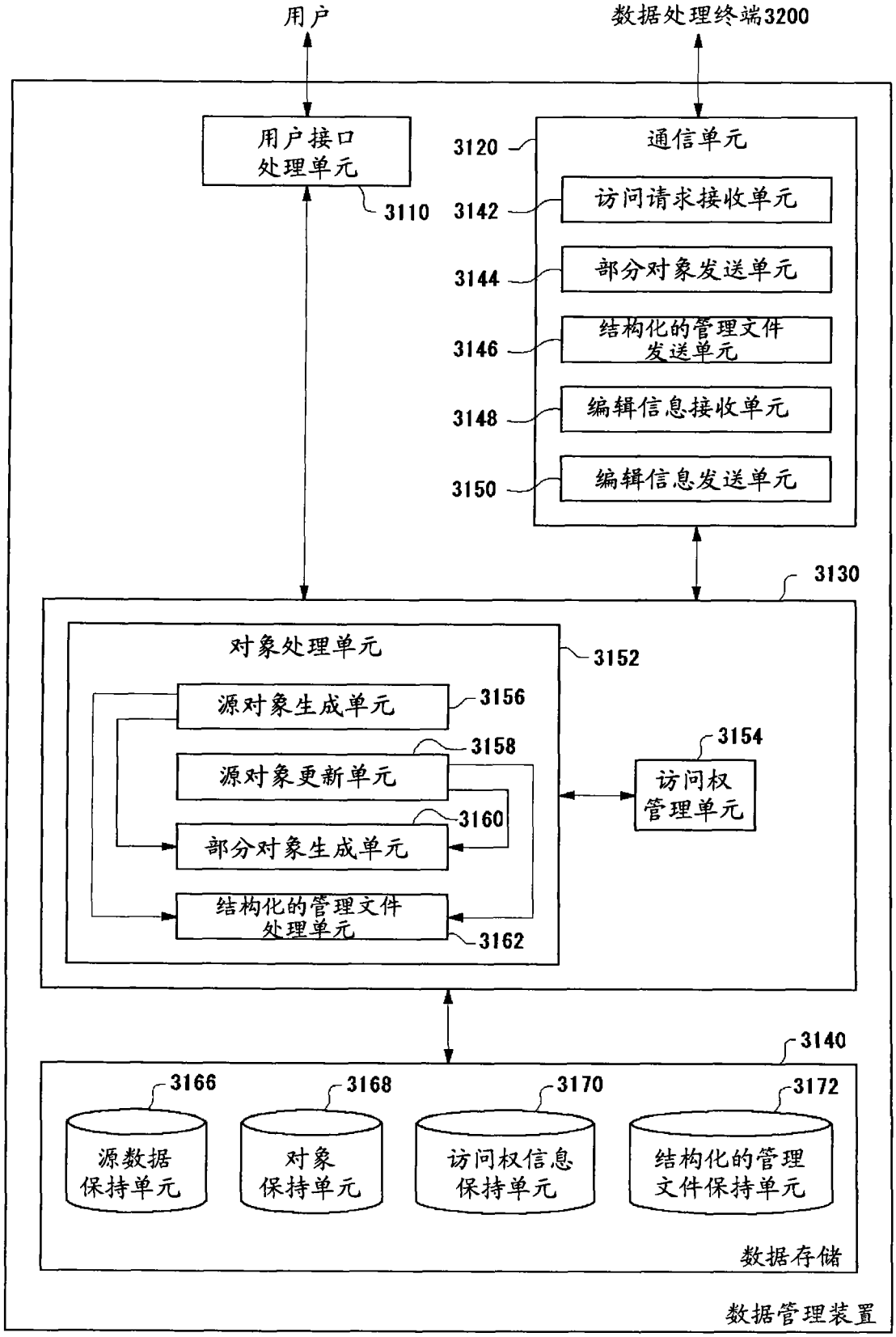
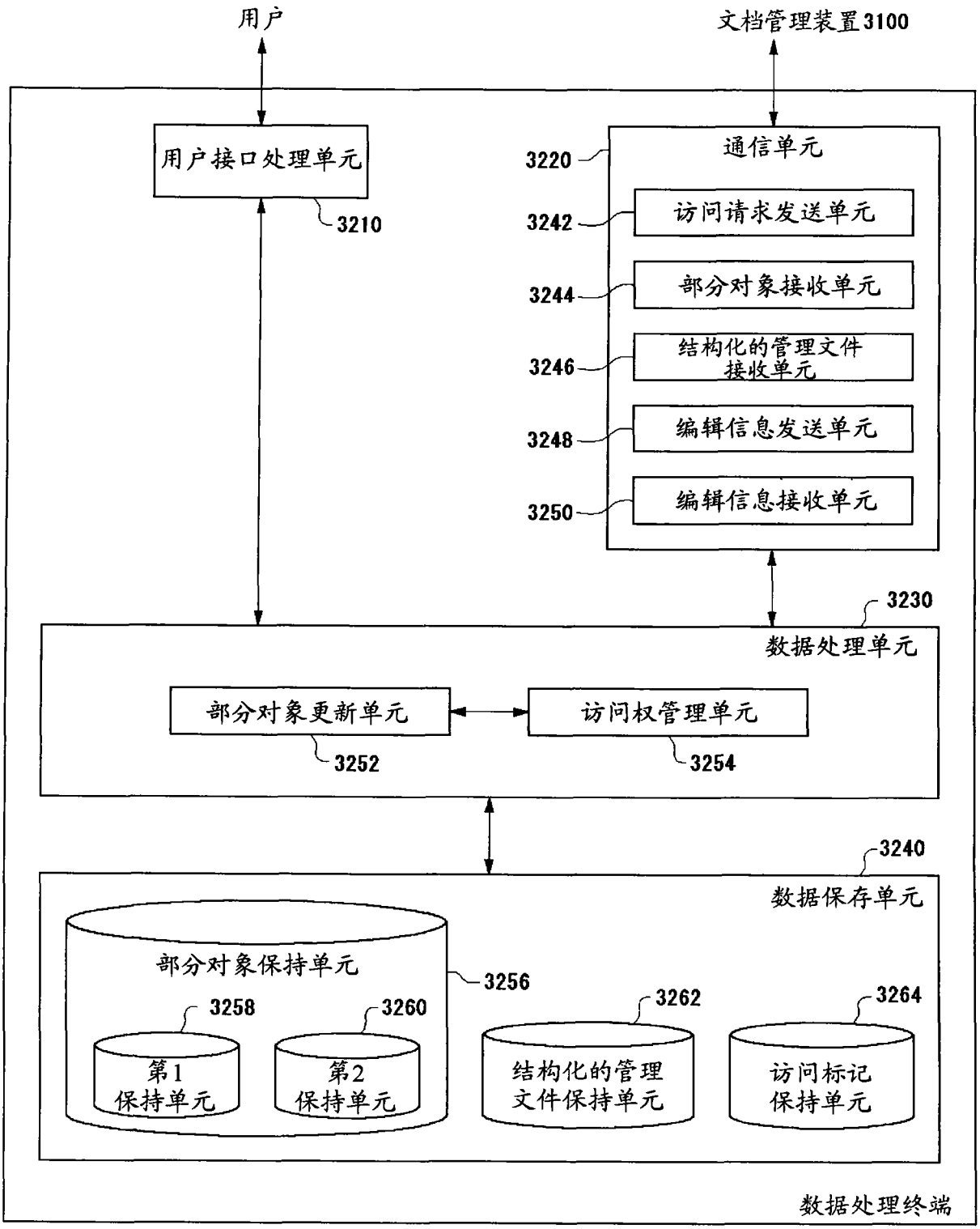

客户端A
客户端B
客户端C
服务器装置
AW
AR
发送 AW (S10)
发送部分对象
(S12)
AR
AW
BW
BR
发送 BW (S14)
发送部分对象
(S16)
BR
BW
CW
CR
发送 CR (S18)
发送部分对象
(S20)
CW
CR
S22 更新处理
发送编辑信息(S24)
更新处理 S26
发送编辑信息(S28)
发送编辑信息(S30)
更新处理 S32
更新处理 S34
CW
发送 CW (S36)
发送 CR (S38)
CR
BR
发送 BR (S40)
发送 BW (S42)
BW

图 31


用户
数据处理终端3200
用户接口
处理单元
3110
3120
通信单元
3142
访问请求接收单元
3144
部分对象发送单元
3146
结构化的管理文件
发送单元
3148
编辑信息接收单元
3150
编辑信息发送单元
3130
对象处理单元
3152
源对象生成单元
3156
3158
源对象更新单元
3154
访问权
管理单元
3160
部分对象生成单元
结构化的管理文件
处理单元
3162
3140
3166
3168
3170
3172
源数据
保持单元
对象
保持单元
访问权信息
保持单元
结构化的管理
文件保持单元
数据存储
数据管理装置

3100

图 32


用户
文档管理装置3100
用户接口处理单元
3210
通信单元
3220
访问请求发送单元
3242
部分对象接收单元
3244
结构化的管理文件接收单元
3246
编辑信息发送单元
3248
编辑信息接收单元
3250
3230
数据处理单元
部分对象更新单元
3252
访问权管理单元
3254
3240
数据保存单元
部分对象保持单元
3256
3258
3260
第1保持单元
第2保持单元
3262
结构化的管理文件保持单元
3264
访问标记保持单元
数据处理终端

3200

图 33

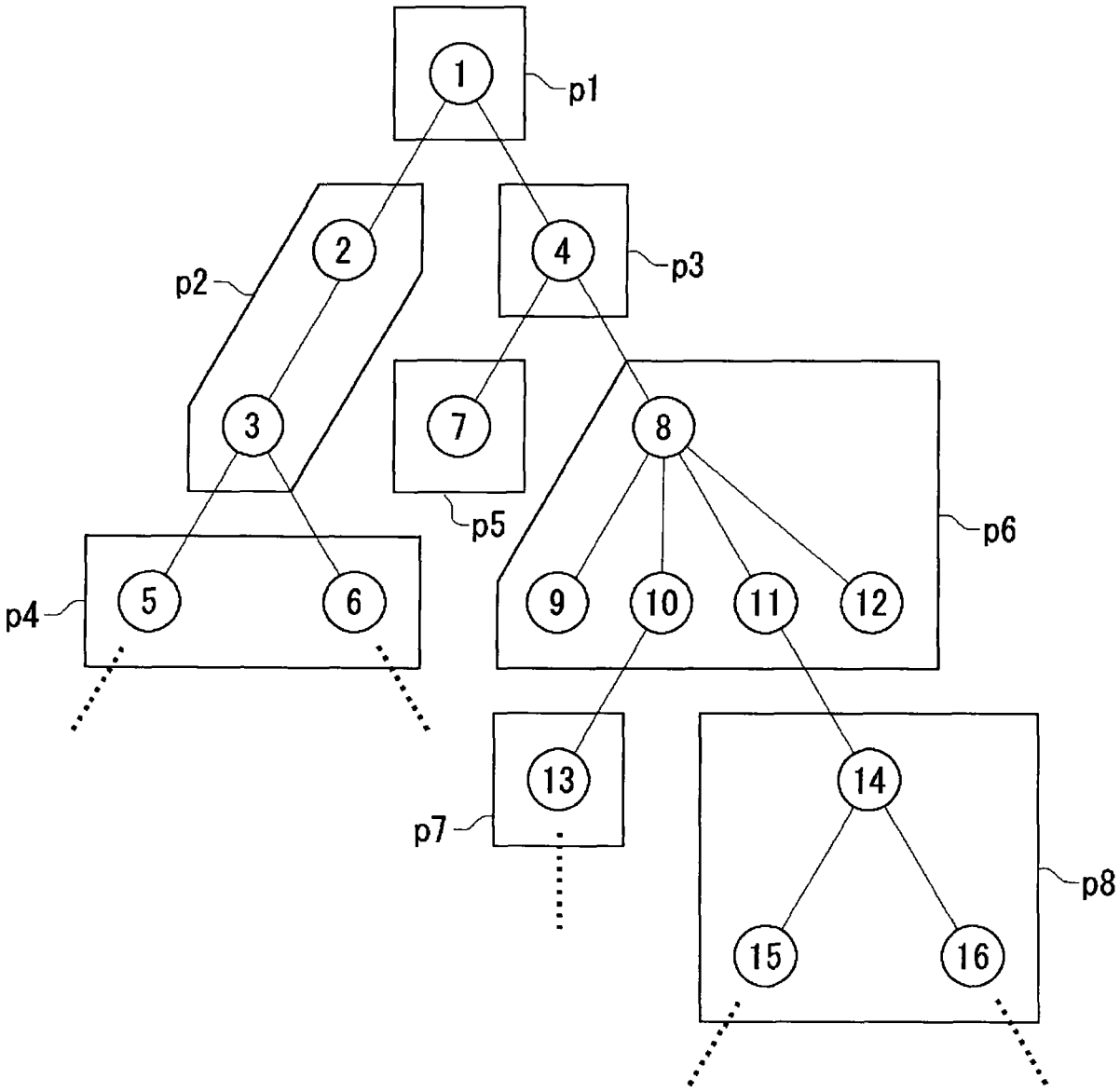
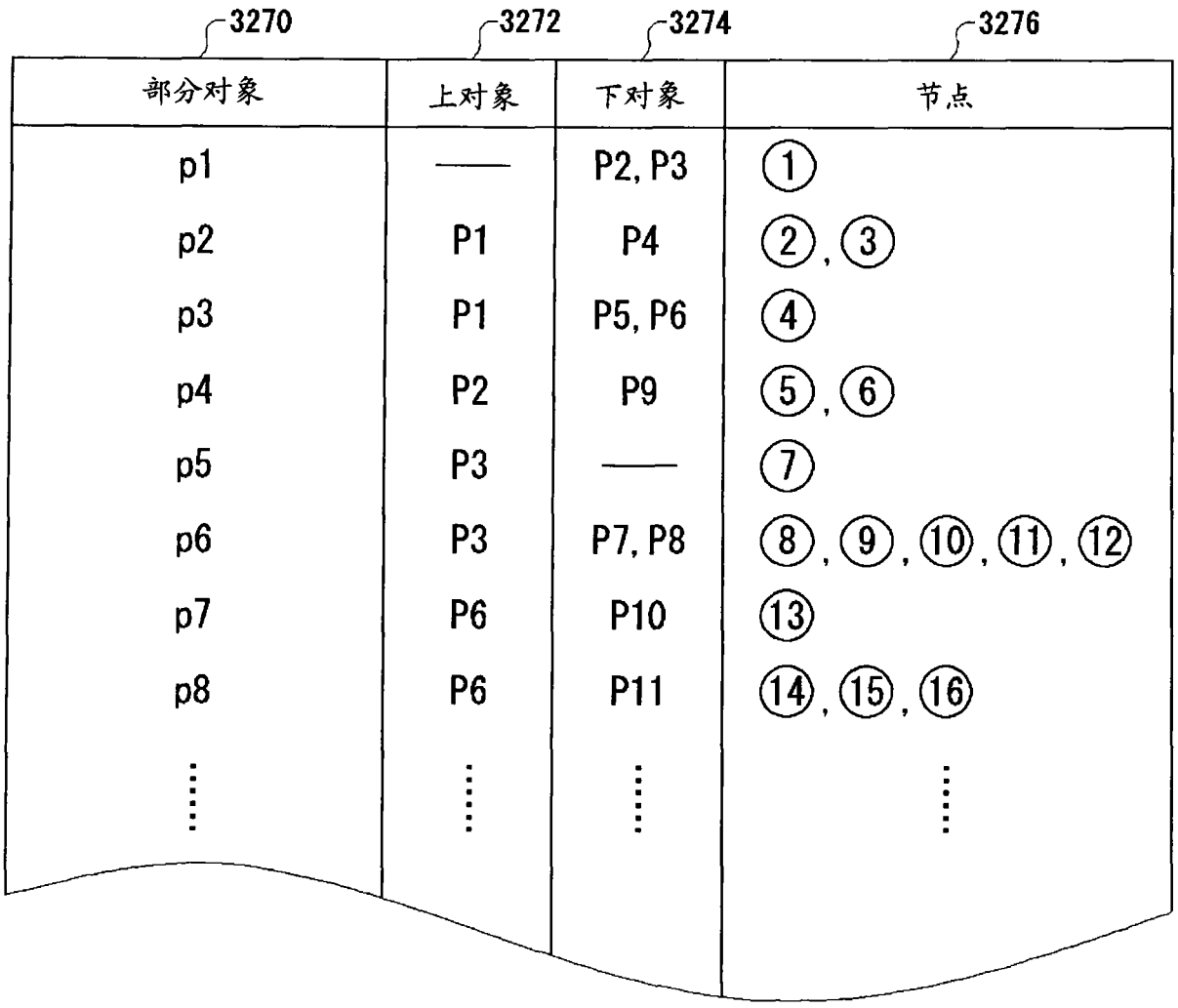


图 34



3270 部分对象	3272 上对象	3274 下对象	3276 节点
p1	——	P2, P3	①
p2	P1	P4	②, ③
p3	P1	P5, P6	④
p4	P2	P9	⑤, ⑥
p5	P3	——	⑦
p6	P3	P7, P8	⑧, ⑨, ⑩, ⑪, ⑫
p7	P6	P10	⑬
p8	P6	P11	⑭, ⑮, ⑯
⋮	⋮	⋮	⋮

3172, 3262

图 35

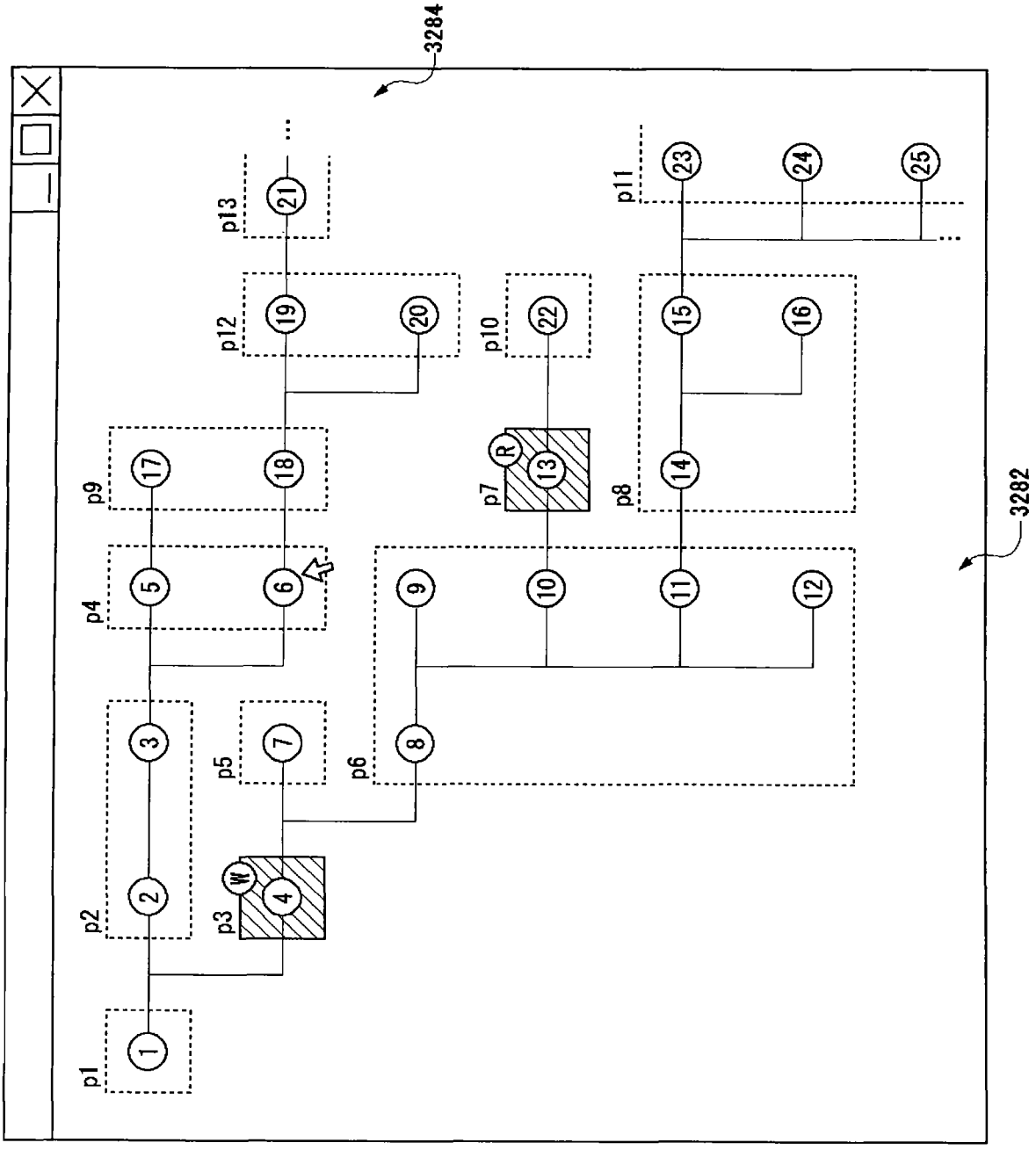


图 36