DOI:10.13656/j.cnki.gxkx.1995.03.017
广西科学 Guangxi Sciences 1995，2（3）：68～70，74

Dexter——超文本系统的一种标准模型
Dexter——A Reference Model of Hypertext

熊瑞萍
Xiong Ruiping

（广西计算中心广西软件新技术实验室　南宁市星湖路32号　530022）
(Guangxi New Software Technology Lab., Guangxi Computing Center, 32 Xinghu Road, Nanning, Guangxi, 530022)

摘要　介绍了Dexter模型的体系结构和基本组成，并着重讲述了超文本系统中的几个重要的基本概念。主要是存储层，运行层及两个接口机制的介绍。

关键词　超文本　链　锚　成员　组合　接口机制

Abstract　As a abstract model and general standard, Dexter has made a contribution not to be ignored and has a position not to be replace. Introduces the architecture and essential compositions, describes some basic concepts of hypertext system, and focus here is on the storage layer, run-time layer and two mechanisms, anchoring and representation, specification which form the interfaces between layers, respectively.

Key words　hypertext, link, anchor, component, composite, interface machanism

1　前言

超文本是80年代开始兴起的一种新的信息管理技术，提供了与传统数据库技术不同的动态信息存贮方式和以链访问数据的信息检索手段，超文本技术的兴起给计算机信息管理领域带来了一场新的革命，并迅速成为计算机界的研究热点，随之各种超文本系统也纷纷涌现，就目前来说，有名的超文本系统有NoteCards、Neptune、KMS、Intermedia及Augment等。这些系统应用在不同的领域和范围，又各有其特定的数据模型和函数集。那么，同为超文本系统，它们所共有的性质是什么呢?如何判断一个系统是否真是超文本系统呢?如何在两个不同的系统间进行比较及分析其异同、或进行交互作用呢?开发一个新超文本系统又有什么标准可循呢?这一连串的问题实际上提出了超文本的标准和本质问题，也就是是否存在一种衡量现有超文本系统并为将来超文本系统的开发提供一种依据的标准。本文所讲述的Dexter标准超文本系统正是这样一种标准。Dexter不是一个实际的超文本应用系统，而是一个抽象模型，它综合了所有现有超文本系统的模式，同时考虑了将来超文本技术的发展而在现有的功能上有所扩展。这种对现存及未来系统的适应性是Dexter成为一种衡量超文本系统的标准的主要原因。

1995-06-23收稿。

2　Dexter模型概述

Dexter模型创始于从事超文本研究的两个小小工作室，其中一个工作室于1988年在New Hampshire的一个名叫Dexter的小酒吧里成立，模型由此得名。在工作室里，超文本系统的共同抽象得以讨论、定形；超文本领域的专业术语也得以完善和确定，以方便人们对各种系统中共同的或不同的概念的理解。需要特别指明的是，作为一种通用标准和抽象模型，为了不致于与具体应用系统相混淆，在超文本系统中普遍使用的“节点”的概念在Dexter模型里均用“成员”这个词来代替。

Dexter将整个超文本系统分为三层：运行层(Run-time Layer)、存储层(Storage Layer)、内部成员层(Within-Component Layer)。运行层与存储层之间的接口关系是表示说明(Presentation Specification)，存储层与内部成员层之间的接口则采用了锚接机制(Anchoring)，如图1所示。

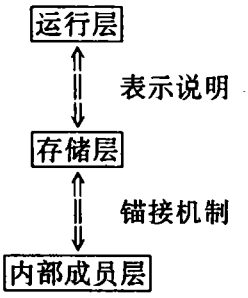

图 1　Dexter 模型的层次结构图

Fig. 1　Layers of the dexter

运行层用于超文本表示，用户在此层与超文本进行会话及交互作用。此层提供了给用户访问、察看及处理网络结构的工具，但由于 Dexter 是一个通用模型，所以在 Dexter 里运行层仅仅是一个将超文本提交给用户察看和编辑的表示机制的一个梗概模型，这种表示机制抓住了超文本系统动态地、交互作用方面的本质，但并不想覆盖用户与超文本交互作用的所有细节；存储层是超文本系统的重点和关键，在此层建立了超文本系统必不可少的基本节点/链网络结构的模型，描述了由被关系链交互连接的包含数据的成员体系所构成的数据库，这里成员对应的就是各超文本系统中的节点。不同的超文本系统节点的内容是不同的，如在 NoteCards 和 Hypercard 中是卡片、KMS 是帧，Augment 和 Intermedia 是文档，Hyperties 是文章，在 Dexter 中，成员只是作为数据的通用载体来处理，在其内部并不建立任何结构模型，因而在存储层里对文本成员和图形成员或其它的成员类型并不加区别，也不涉及成员内的结构；内部成员层则专门考虑超文本网络中成员内部的具体内容和结构细节，超文本中成员的内容和结构是不受限制的，可以是文本、图片、声音、动画等各种媒体的数据类型，作为一种通用标准和抽象模型，不可能也不必要覆盖所有的数据类型，因而对于此层，Dexter 特意不加完善，而是将它作为一个外部处理过程，留待超文本系统的开发者自己去定义。

Dexter 模型对超文本系统的贡献还在于所提出的两个接口机制：表示说明和锚接。通过表示说明机制，有关一个成员网络是如何显示给用户的情况能够编码送到存储层的超文本网络中，因而，成员表示给用户的方式可以不仅仅是一个执行显示的特定超文本工具的函数，也可以是一个成员自身及到该成员的访问路径（链）的性质。如图 2，有一个从基于计算机的训练超文本系统中产生的动画成员，这个动画成员可通过两条链成员进行存取。当跟踪教师编辑链时，以可以修改的编辑方式取出动画；当跟踪学生链时，以运行方式取出动画。为区分这两种情形，表示说明机制将运行层所需要的显示信息编码，送入存储层的网络中。通过这种机制，实现了存储层与运行层的通讯而又保持了两层各自的独立性。

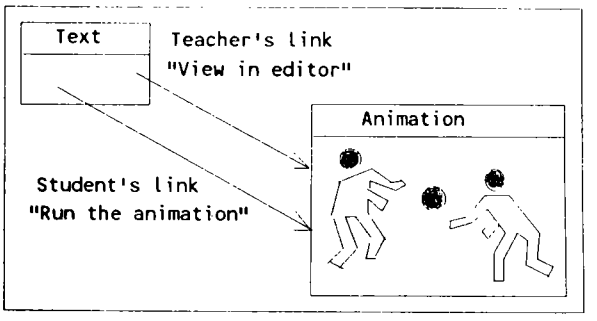

图 2　成员自身与访问路径对表示说明的需要

Fig. 2　Illustration of the need for presentation specifications on the access path (i. e. , links) as well as on the components themselves

另外一种接口机制是锚接。锚接是一种提供维护存储层和内部成员层之间的独立性的锚接函数的机制，通过这种机制，既维持了存储层网络与内部成员层数据之间的独立性，又能够较好地查找与处理成员的内容和结构。它的实现方法将在下面详细论述。

Dexter 的重点是放在存储层、运行层及其两种接口机制上，下面分别介绍这两层。

3　存储层

存储层由一个成员的有限集合与两个函数组成，在这里，成员可以是节点，链或一个包含若干节点和链的组合。节点是信息数据的载体；链是反映两个节点间关系的一种实体，最基本的表示就是两个或更多端点所组成的序列，其中每一个端点都与超文本中的一个成员有关；组合由若干成员构成，当一个组合包含有另一个组合时，所建立的组合成员层次关系被限制为一个有向非循环图（DAG），这意味着一个成员可以是多个组合的子成员；不管是直接的还是间接的，组合都不能包含自身。两个函数是分解函数（resolver）和访问函数（accessor），它们共同负责成员的检索（如将成员说明映象到成员自身）。

存储层里每个成员都有一个标识符，称为 UID。UID 是超文本中的一种基本的直接寻址机制，也就是说，通过一个给定的有效的 UID，即可找到一个相应的成员，这也就限定了 UID 在整个超文本系统中具有唯一性和全局性，利用 UID 可为在超文本中寻址成员提供一种保护机制，但作为一种基本寻址机

制，直接使用 UID 来寻址限制太多。如给一条语句建一条链是可能的，但这条链所说明的目标语句可能不存在或在文档编辑期间已被改变，因而链不能仅仅依赖于一个特定语句的 UID 去寻找目标语句。相反，当链被跟踪时，表示说明必须被分解为能够用来访问正确成员的一个 UID 或 UID 的集合。因此，存储层提供了一种间接寻址方式：

$$\text{成员说明}\xrightarrow[\text{转化为}]{\text{分解函数}}\text{UIDS}\xrightarrow[\text{检索}]{\text{访问函数}}\text{被说明的成员}$$

也就是说，分解函数负责将成员说明转化为 UID，UID 被送给访问函数以检索被说明的成员。需要注意的是：一个给定的说明可能无法被分解为 UID（如被说明的成员不存在），但对于每一个成员来说，至少有一个说明可以分解为它的 UID。

另一种间接寻址机制是存储层与内部成员层之间的接口机制：锚接。通过这种机制，保持了超文本网络自身与成员内部内容和结构细节的独立性。简单地说，锚是一种间接寻址实体，它由两部分组成：锚值和锚 id。锚 id 是一个标识符，在成员的工作域内唯一说明锚，是一个不变量；锚值则是一个说明某些位置、区域、项或成员内部子结构的随机值，即锚值是会变化的，且这种变化是系统所无法预料的。如当一个成员被改变时（如在运行层里被编辑），那么内部成员应用程序会改变锚值以反映成员内部结构的改变或锚所连接到的点、区域或项的内部成员的移动。

锚 id 机制和成员说明机制结合起来就可以提供一种定义链端点的方法，这是通过一种称为说明符（Specifier）的实体来实现的。说明符的组成如下：一个成员说明（Component Specification）、一个锚 id、两个附加字段：方向字段（direction）和表示说明字段，其中方向字段反映的是链的起始和指向，可为以下四个编码之一：Form（链源）、To（链目标）、BIDIRECT（即是链源又是链目标）、NONE（即不是链源又不是链目标）。从而链又可定义如下：链是两个或更多说明符组成的序列。在大多数模型里，一条链允许有多个说明符，但必须至少有一个说明符的方向域是 To 或 BIDIRECT，也就是说链必须至少有一个目标。

以上是存储层中定义的数据模型。此外，存储层还定义了一个操作集，可用来访问和修改超文本，如在超文本中增加一个成员；从超文本中删除一个成员；修改一个成员的内容或辅助信息，等等，在这些操作中，一些可用来检索给定 UID 或能够分解为 UID 的说明符的成员，一些则是帮助决定网络结构的交互连接性的。

按照 Dexter 系统的存贮层模型，一个超文本系统应遵循以下约束条件：

1）访问函数从 UID 到成员的映象是可逆的；

2）分解函数必须能产生有效的 UID；

3）成员与子成员之间不存在循环关系，即成员不能是自身的子成员；

4）成员的锚 id 必须与分解成此成员的链说明符的锚 id 一致；

5）超文本必须是连接一致的（即当删除一个成员时，说明符能分解成此成员的链也必须被删除）。

4 运行层

运行层是用户与超文本系统交互作用的地方，涉及到的主要基本概念是实例（instantiation）和会话（session）。实例是提供给成员的一种运行缓冲区，存放成员的备份，用户可以对它进行观察、编辑，然后被改动过的缓冲区内容又写回到存贮层中。象成员和锚一样，实例也具有唯一的标识符，称为 IID。一个成员可以拥有多个实例，会话是保持成员和它们的各实例之间的映象的轨迹的一种实体，实际上也就是用户与超文本系统的交互活动。会话实体包括：正被访问的超文本、当前实例的 IID 到它们所对应的超文本中成员的映象，操作历史（history）、运行分解函数（Runtime resolver）、实例函数（instantiator）和实现函数（realizer）。操作历史是从最近打开会话操作开始执行的所有操作的序列；运行分解函数是前面所讲述的存贮层分解函数的运行版本；实例函数是运行层的核心，它的功能是在传送到实例的表示说明和附加到被实例化的成员上的表示说明之间作出判断，在目前的模型中，这种判断尚不是很明确。

实例函数是表示成员操作的核心，表示成员操作又是跟踪链操作的核心。跟踪链产生实例的 IID 及包含在实例内部的链标识，然后表示处于所有链的目标端的成员。

与实例函数相反的是实现函数，该函数产生实例并返回一个新成员，此成员反映出实例的当前状态，这是将编辑后的实例写回缓冲区的基本机制，实现函数所产生的成员作为一个变量送到存贮层修改成员操作中，用编辑过的成员代替原来的成员。

5 实际超文本系统与 Dexter 模型的一致性判断

如何判断一个超文本系统是否真的称得上是超

（下转第 74 页 Continue on page 74）

端点的显示，这里的端点指的是被组合的封装结构所指向的成员。

如何将存储层的成员和内部成员层的数据对象统一起来，及如何将成员和组合的内部统一起来是在开发开放式的基于 Dexter 模型的超媒体系统时能够支持成员内容的大范围浏览所必需要解决的问题。下表对成员内容作一个小结。

成员内容三要素

结　构	类型/位置	定　义
·原子型 ·非结构化集合 ·结构化集合 —检索清单 —关键字表 —树 …	·数据对象 —在成员内部 —在成员外部 ·成员 —具有严格类型的 —无限制的	·封装在此成员内 ·全局可见

5 结束语

Dexter 模型虽然已成为超文本系统设计的一个标准。但作为一个初步的抽象模型，还有许多方面需要完善，许多概念需要清理与扩展。本文所论述的问题仅涉及到链、锚、成员等主要超文本对象类的设计，可作为开发基于 Dexter 的超媒体系统时的参考。

参考文献

1 Kaj Grdnbaek Randall H. Trigg: For a Dexter-Based Hypermedia System. Communictions of the acm, 1994, (2): 37.

2 Frank Halasz, Mayer Schwartz: The Dexter Hypertext, Communications of the acm, 1994, 37 (2).

3 John J. Leggett, Johnl · Schnase: Dexter with,open eyes. Communications of the acm, 1994, 37 (2).

(上接第 70 页 Continue from page 70)

文本系统，它与 Dexter 标准模型是否一致？一种方法就是将实际超文本系统中的数据类型和函数映象到 Dexter 中的每一种数据类型和函数上，且充分论证，对于 Dexter 模型中的每一个抽象值在实际超文本系统中都有一个有效的表示，换句话说就是，在模型中可表达和可实现的每一件事在实际系统中都必须是可表达和可实现的。

但事实上，由于考虑到将来超文本系统的发展，Dexter 模型的功能远远超出目前超文本系统所必需的，主要表现在以下几个方面：

·允许多路链接和链到链的连接

·具有组合的概念

·具有同时拥有显式源和目标的链

在这里，要解释的是组合这个概念，这里的组合不是指节点，链等原子的简单组合体，在 Dexter 模型里，这种组合体和节点，链一样，都属于基本成员。所谓组合是指包含基本成员及一些相关的描述成员性质而不是内容的成员信息，同时还包括一个能索引成员的锚系列、将成员显示给用户的表示说明和一系列任意数目的属性－值对。属性－值对可将任意性质附加到一个成员上，这种组合的概念是目前已有的各种超文本系统所不具备的。

6 结束语

Dexter 模型目前尚处于开发的早期阶段，许多方面还未能很好完善，但功能及结构定义已能满足并远远超过现存的超文本系统，一些概念，如 n 元链和组合成员的构造就是为了适应将来超文本系统的发展和设计的。因而，作为衡量及判断超文本系统的标准、Dexter 模型可以说是一种理想的模型，对于指导我们开发超文本系统具有重要的指导意义。

参考文献

1 Halasz F., Schwartz M. The Dexter Hypertext. Communications of the acm. 1994, 37 (2).

2 Aksyn R., McCracken D L, Yoder E A, KMS: A Distributed Hypertext for Managing Knowledge in Qrganizations Commun, ACM 31, 1988, 7.

3 Shneiderman b: Hypertext on Hypertext, Addision Wesley, New York, 1989.

4 Panei ∩ T,: Interchanging hypertexts. In Proceedings of Hypertext 89, 1989, 5～8.

5 Dick L H., Bulterman C A, Guido Van Rossum: The Amsterdam Hypermedia Model, Communications of the acm, 1994, 37 (9. 2).