《计算机教学管理（CMI）系统规范》

实践指南

（草案）

教育部教育信息化技术标准委员会

2002 年 3 月

目录

第 1 章 概述

1.1 标准概要

这个标准不是定义如何设计与实现计算机教学管理（CMI）系统，而是从四个方面论述了实现计算机教学管理系统互操作性（interoperability）的指导方针。包括：

- 如何管理学生的学习活动，主要是上课活动
- 如何在一节课上课前后，在课与 CMI 系统之间交流信息
- 如何让原先在某种 CMI 下运行的课程，可以在另外的 CMI 系统下工作
- 如何保存课堂评估数据

上述每一方面都会在多种环境下讨论，包括局域网环境和因特网环境。因为互操作性涉及到各种文件，所以标准中也描述了这些要用到的文件的格式及内容。

1.2 背景

基于计算机的教学与培训课程（CBT）多使用写作系统进行开发，这些开发的课程往往需要使用计算机教学管理（CMI）系统进行管理，以发挥 CMI 的功能和优势，但是在以前，一般一个 CMI 系统只能管理单一厂商的 CBT 课程。而用户可能会使用不同的写作系统开发的 CBT 课程，比如：

- 随新购软、硬件奉送的 CBT 课程
- 从使用另一写作工具的供应商处购买的课程
- 决定使用其他写作系统设计新的 CBT

这些课程与现行的 CMI 系统就可能不兼容，有很多原因会促使一个单位更趋向于使用单一的 CMI 系统，而不是为不同的 CBT 课程配备不同的 CMI 系统：

- 教师已经熟悉了现在使用的 CMI 系统，再学新的系统需要花很多的时间，这会影响新课程尽快使用，还会为此付出培训费用。
- 尽管一系列课程的内容不同，但是有必要使学生有一个整体的统一的界面感觉，CMI 的学生界面是形成统一感觉的重要组成部分。
- 维护两种不同的 CMI 系统比维护单个 CMI 系统要复杂。
- 现有的 CMI 系统有着与新课程相适应的 CMI 所没有的特点和功能。
- 希望将用其它写作工具设计的新课加到现有课程中，整个课程只用一个 CMI 系统
- 要求从现有的课程中各取出几节课组成一门新课程，如果这些课程各自需要特定的 CMI 系统，则几乎无法满足这一需求。

因此存在这样的需求，即将不同的 CBT 课程在同一个 CMI 系统中使用。这样的需求如果没有一个计算机管理教学（CMI）功能的标准集合和与之相匹配的 CBT 功能集合，是难以实现的。当然，互操作性标准同样也为课程内容开发者提供了便利。一旦知道他们开发的内容可以在兼容的 CMI 上任意传递，内容开发者们就拥有了巨大的潜在市场。

另一种需求是将课程从一个 CMI 环境搬到另一个 CMI 环境：

例如，课程已经开发完成了，从厂家或销售商的手中交给了使用另外一种 CMI 系统的某个单位。如果手工的将一个新课程转换到一个现有的 CMI 系统，需要输入上百次课程名称，复制所有的次序信息，要花费很多人时。但是如果对于课程内容和结构能有一个标准化的描述，就可以用最少的人力将一个新课转入 CMI 系统。

或者是课程不是用 CMI 系统开发的，而是用工具开发的，需要将课程设计导入 CMI。

能够设计课程的工具很多，最常用的是任务分析工具。如果课程设计工具能够输出关于这个课程的标准描述，那么 CMI 系统就可以利用这个描述将新课集成进来，这可以节省大量重新录入和输入数据的时间。

还有可能是有一个新的、功能完备的 CMI 系统，需要将现有的所有课程都集成到新系统中去。

课程设计隐含地定义了课程的使用行为。课程搬迁，不只是搬迁教学材料，还必须保留课程所设计的行为定义，以保证课程的理念和功效也随之转移。

比如一门课程被设计为按顺序讲授，教学内容逐步呈现，学生必须学完了已经呈现的才能看到下面要学的。但是如果改变 CMI 系统后，教学材料一下子全呈现给学生，成为可以自由读取的学习方式，那么学生就可能会跳过某些章节，因为他们认为自己已经知道了，并因此错过一些教学内容。在这个例子里，课程的可靠性因为没有保存课程行为纪录而被削弱了。反过来，假设一门课程原先设计为教学内容可以自由获取的，学生只要通过一组测验，就被认为完成了学习任务。但是在改变 CMI 系统后，CMI 要求学习进程顺序进行，学生就要在课程上花更多的时间，被迫服从顺序而去学习原本可以跳过直接考试的内容。在这个例子里，学生接受课程和完成的时间也因为没有保存课程的行为要求而受到影响。

还有一种需求来自方便地分析学生数据。可以进行学生数据分析的工具很多，例如，电子表格、数据库等等。如果不同的课程收集了学生数据，必须使用不同的分析工具进行分析，就会给使用者造成相当大的不便。

1.3 目的和意义

本标准说明了 CMI 系统和 CBT 课件应该具备的性能，以便：

- 同样的课可以在不同的 CMI 系统中工作，
- 分别开发的课能够结合而成为一个 CMI 系统的课程，
- 课程可以从一个 CMI 系统搬到另一个 CMI 系统，
- 方便分析学生数据。

下面给出了实现这些目标所能带来的好处：

1 允许同样的课在不同的 CMI 系统中工作

这使得教师或课程开发者可以使用不同来源的课创建一个课程。例如他们可以通过因特网收集全球的有关课程，放在一起，排好次序，再用单个 CMI 系统加以管理。

这也使得开发者可以充分利用原有的课件，即使早期的课是用不同的开发系统设计的，只要遵循标准，它们就可以与新开发的课程单元一起使用。

现有的课程也能很容易的被修改和扩展。依照这一标准的课可以与现有的课程结合，同样可以在现有的 CMI 系统中工作。

2 允许课程从一个 CMI 系统搬到另一个 CMI 系统

这个目标的好处可以用下面三个例子解释。

例 1：假设课程是老师用另一个 CMI 系统开发的，你可以使用他的课程内容，但是不能用他的 CMI 系统，如果这个老师教学内容的设计遵从这个标准，你就可以将这些资料导入你的 CMI 系统。

例 2：课程是一个厂商开发的，你买了这个课程，但是不想要他的 CMI 系统。如果这个课程是按照这个标准设计的，你就可以将课程内容导入你的 CMI 系统。

例 3：你用 CMI 系统 A 管理了若干课程，你很熟悉这个系统，而且很喜欢它，现在你又得到了一个新课程，必须在 CMI 系统 B 下运行。因为不想学习新系统，也不想同时用两个系统，所以你想将系统 B 中的课程导入系统 A 中，这样你就可以用自己的 CMI 系统，如果新课程和你的 CMI 系统都是按照这个标准设计的，你就可以用你喜欢的 CMI 系统运行所有的课程。

3 方便分析学生数据

同样，下面的例子可以清楚解释这项目标的价值。比如你的课程收集了学生数据，为此，你设计了一个电子表格来分析这些数据。现在在你的课程中用了一个新课件，但是它存储数据的方式不同，所以不能将这些数据导入你的电子表格，你只能手工录入。本标准可以防止这些问题发生，如果各种数据都以标准格式保存，电子表格就可以输入和分析这些数据，而不必手工录入，也不用修改电子表格。

第 2 章 CMI 概述

计算机教学管理(CMI)系统管理学习环境中的课件和学生。这一章描述 CMI 系统需要具备的基本和高级功能，其目的是定义 CMI 标准的范围，了解本标准所讨论的 CMI 的内涵，可以加深对标准的理解。

在讨论学习活动时要用到许多术语，对于术语可能有不同的解释，为此，在本标准草案的第二章给出了这些术语的定义。

本章要用到的术语有：

- 可分配单元
- 块
- 课程
- 课程体系
- 层次
- 课

CMI 系统不仅仅是对 CBT 材料的计划安排，它还能管理在线(CBT)和离线的教学活动和测验。通常 CMI 系统有 5 个组成部分：

- 课程结构开发；
- 测验；
- 学生注册；
- 学习布置管理；
- 数据收集和管理。

2.1 课程结构开发

CMI系统的核心是课程开发部分。为了让CMI能够管理学生数据、安排学生学习，就必须定义学习材料的结构和层次。

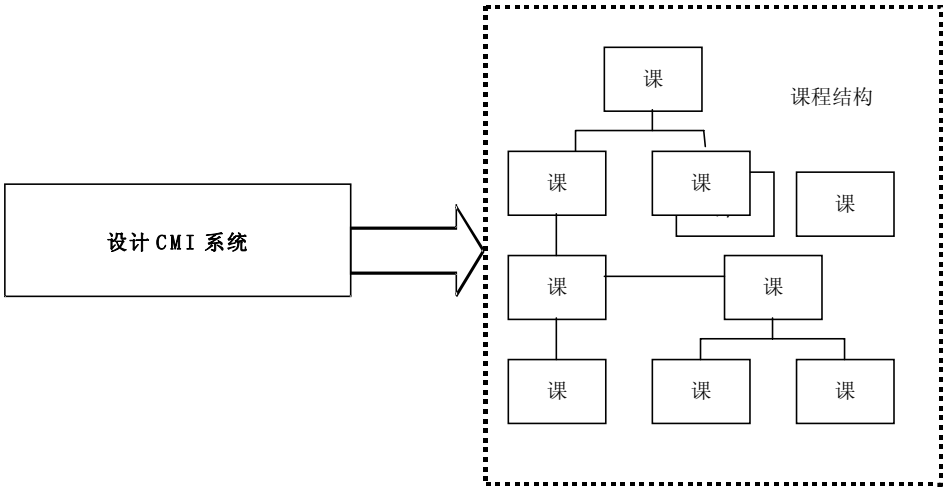

图1　课程结构

课程结构工具要能定义课和它们的属性。这些工具作用于最小的可布置、可跟踪单元（即一节课）。

每节课的属性包括：

- 相关的学习目标和分类；
- 测验标识；
- 允许重试的次数；
- 课类型 (在线/离线)；
- 课布置规则；
- 如果所有重试失败，是否需要教师的介入；
- 设备、教室、教师、学习用品（如练习簿）；
- 辅导策略及补习课。

课程结构工具通常还包括定义学习目标的方法，包括目标标识、目标陈述。

课可以由CMI系统安排而有不同的上法。也就是说，课可以依据CMI 系统传来的信息而表现不同的行为。比如，课的行为可以基于学习次数、学生态度、过去的表现或课程开发标准来决定。

灵活的 CMI系统可以安排任何一种CBT开发系统开发的课，也可以安排来自几个不同CBT开发系统的课。学习每节课的数据都被返回到CMI系统用于学习追踪和报告。

每节课都有有效性指示器，一旦学完这节课内容，学生再看这节课内容，就是将其作为参考资料浏览。

课程结构提供了一种将课分组的方法，以方便教学次序安排。根据课的层次结构，课程开发者可以定义某组课或单个可分配单元的先修课与后继课。

示例

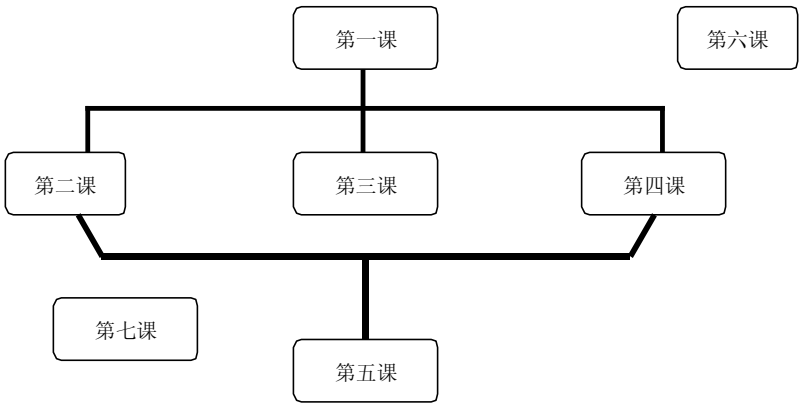

图2　课的顺序

在这个例子中，共有七节课，第一课必须先学，第二、三、四课学习次序随意，但是必须要在第一课之后第五课之前。第六、第七课在任何时间学都可以。

在管理课程资源时，能否定义复杂课层次非常重要，通常需要定义多重路径。除了课层次以外，层次概念也可以作用于由课组成的块，比如将上面课层次作为块1，含有块和课的层次图可以如图3所示：

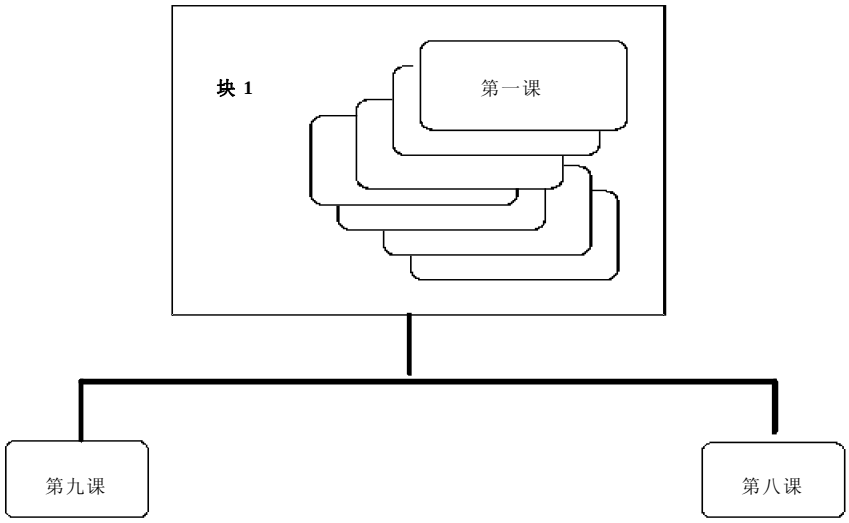

图3　课与块排序

资源可以被CMI系统定义和管理。资源可以多次使用（如设备）或消耗（如纸张），需要再定购。教室也是资源的一部分，需要根据课来安排和使用。像教师一类的资源还需要相关的资格条件。CMI 可以很容易的维护现有需求的资格条件和状态。

CMI系统应该允许由外部机构安排课序，如教员、或其他排课序系统，这一功能可以通过定义两个课程附加属性：资源限制（resource-limited）、代理安排（agency-scheduled）来实现。

资源限制课程由CMI系统安排。代理安排课程由外部机构安排，在完成的时候需要通告CMI系统，这种通告可以是学生完成CBT课后系统的确认，也可以是教师向 CMI提交的完成表格。

其他功能可以由课程结构部分来提供。CMI系统可以支持跳班或改变培训时间，这对于最大化利用资源很有用。日历提供了确定假期、会议、课程进度控制的方法。学生学习自动升级算法保证了课和课程之间平滑地过渡。

课程结构中的测验点的概念，相对于在每节课结束后的相关测验，提供了更大的灵活性。它允许测验在一组课完成后进行，使学生在继续课程前可以进行补习。

对于从数据交换文件中装载课程结构数据库的支持，提供了在不同的CMI系统之间，任务分析系统与CMI系统之间的传递课程数据的方法。

2.2 测验

测验部分主要是用于在线和离线测验的开发和管理。测验可以由下面的系统处理：

- CMI 系统 ；
- 分离的测验系统(离线)；
- 传统的 CBT。

每个系统都必须能将测验结果报告给CMI系统。

测验常常为课的一部分，作为CBT系统的一项功能单独处理。在有的系统里，测验的开发可以由CMI 进行，这并不排除使用CBT系统进行测验的开发。

测验的类型包括掌握程度测验、行为测验、态度测验。如对态度的调查问卷可以使用测验系统的题目分析工具进行评估，而不需要其他的评估工具。测验可以在线管理(通过CBT系统)或者是离线管理(如, 纸或观察表现)。离线测验的数据必须输入到CMI系统，如通过扫描仪扫描或用在线表格提交来完成。

测验由测验题目组成。测验题目与教学目标有关。CMI系统支持标准参照测验和规范参照测验。对于掌握程度的测验，题库和试卷库都是很有用的。灵活的测验评分策略相对于标准百分比测验要增加：

每个教学目标对应的题目个数

- 在一个测验中所包含的教学目标个数；
- 题目权重 (对意见调查卷特别有用)；
- 关键题目(通过测验所要通过的题目)；
- 关键目标(通过测验所要通过的目标)。

测验可以是课前测验或课后测验。通过课前测验，就可以获得该节课的学分，这意味着学生已经掌握了这节课的内容，无需再上这节课。如果测验不合格，就要强迫学习这节课的全部内容。

测验部分可以为题目分析收集数据，如果测验是通过传统的CBT课程管理的，则需要通过一定机制，将结果输入到CMI系统中作题目分析。

2.3 注册操作

学生注册操作可以在课程开始之前录入学生姓名和有关数据。有时也称为登记。这种注册不仅要登记学生的基本数据（学号和学生姓名）及所选的课程，也要登记学生的一些个人信息。

CMI系统也可以进行批量注册和/或自行注册。批量注册可以一次完成整个班级或整个学习中心的注册。自行注册（或自行录入）是学生在没有教师/管理者的介入下自己注册课程的方法。

CMI系统还要提供对学生退课的支持。退课有两种做法，一种是将学生从该课程中删除，但是保留学生的行为数据，做以后分析用；另一种做法是在删除学生的同时删除与该生所有相关的数据。

标准报告向教师和课程开发者报告日常管理情况。基本报告包括：

- 选课学生名单；
- 当前的课程安排(在一门课中对于所有学生的当前安排)；
- 资源利用(正在使用的资源和可以使用的资源)；
- 学生表现记录 (学生上完每节课后基本的表现数据)；
- 课程导航图(图形的或描述的)。

管理者或教师可以将学生数据记录由CMI系统传入或传出，如果教学活动在地理上是分散的话，也可以在不同站点之间传送。

2.4 学生学习管理

这一部分提供的功能包括：

- 管理者和教师能够监控每日的学习情况，在需要时介入；
- 布置管理器基于一套规则（预先确定的或用户定制的）控制学生学习；
- 启动课程的标准步骤，使 CMI 能够启动来自不同 CBT 的课程；
- 学生登录功能控制和管理学生的进入、维护学生可访问的数据记录，显示学生的当前学习安排。

布置管理部分可以执行日常的课程安排和学习记录功能。它可以作为学生的主要界面，使用不同CBT系统提供的课程材料。

在这里，可分配单元（课）的概念显得尤其重要。可分配单元是CMI系统进行布置和跟踪的最小单元。一个CBT系统可以使用更小的课程材料单元，但其学习结果在返回给CMI系统时必须对应这些可分配单元。

布置部分支持教师布置、系统布置，也提供学生对于系统的访问。这在下面标题中讨论：

- 教师/管理者功能；
- 系统教学布置管理；
- 学生登录。

2.4.1 教师/管理者功能

除了课程安排和结果记录，还有一些帮助教师/管理者日常工作的功能，包括：

- 确认布置；
- 重新布置；
- 教师评估。

要提供功能支持管理者或教师来判断一节课是否完成，完成状况如何 (例如, 通过, 不及格, 没完成)，如果需要，还可以给出评分。

重新布置允许管理者或教师改变学生要学习的课(将系统原定的布置改变到该课程内的另一课）、或者改变学生要学习的课程。

由于教师在培训和教学中扮演者重要角色，CMI系统必须要能够支持方便、灵活的教师评价信息的输入。教师要能够很容易的建立和修改教师评估表格。相关的数据包括：

- 评估者编号；
- 要评估的任务；
- 评估日期/时间；
- 任务级通过/没通过判定；
- 每一目标通过/没通过判定；
- 目标中每一道题的等级；
- 教师评语；
- 修改评价的教师号，日期，时间。

当学生在某一节课经过了最大重试次数还未能通过时，应该允许学生能继续学习这门课。有两种解决办法：一是标记学生本节课合格，二是将学生重定向到其他课。

2.4.2 系统教学布置

所有的CMI系统都提供教学任务分配，引导学生从一节课到另一节课。

教学布置管理功能有：

- 记录学生在当节课的进展；
- 决定学生下次学习安排；
- 启动学习安排。

较复杂的布置管理系统依据学生的个人需要决定学生的学习内容。它提供了裁剪课程满足学生需要的方法。比如，系统至少可以跳过某一节课直接进入测验。此外，基于课前测验，也能提供个性化的布置策略。

更复杂的功能是允许布置管理系统基于一定的标准选择课，比如根据：

- 学生过去的表现 ；
- 个人数据，如语言、学历；
- 学生的偏好。

高级布置管理系统的另一组成是资源分配模型。这个模型基于现有的使用情况，最大化资源利用率。资源分配系统能够发现由于资源缺乏，布置给学生的课不存在的情况，及时停止课程进程。资源分配算法使用资源权重(比如，先布置或最后布置最关键资源)和过去资源可用情况的数据来决定课的布置规则。

布置管理系统还维护学生表现的数据记录，这些数据记录包括以下信息：

a）课的完成信息；

b）课进展数据：

1）重试次数；

2）上课的时间；

3）通过/没通过状态（按目标号）；

4）每节课的通过/没通过状态；

5）每节课的分数；

6）每节课的状态（停滞、进行、完成等）；

7）开始数据和完成数据；

c）课进行中的当前资源；

d）管理信息：

1）个人数据；

2）升级情况；

3）当前的教室/教师；

4）课程完成数据(如, 通过/不及格，日期)。

2.4.3 课的启动

CMI系统必须能够定位和启动对学生布置的学习内容。这就是启动功能。

2.4.4 学生登录

CMI系统为学生提供了进入学习材料的唯一入口。学生登录的最主要功能是使学生进入其培训程序。在学生登录后，CMI系统可以显示对学生的布置安排，然后启动相关的布置(如果它是基于计算机的课或测验)。

通过学生登录，对课程材料提供了一级安全保护，向学生显示的只是他获准访问的数据。这些数据包括：

- 过去表现的历史纪录
- 目前的教学安排
- 目前在课/课程中的位置
- 课程进度图，示意课程哪些部分已经学完，哪些还未学
- 可能的或选安排

学生登录系统为学生提供了从教师和其他学生那里接收和发送邮件的地方。它还能提供注释工具。

如果教师还未对该生进行注册的话，学生登录还支持学生自行注册。

2.5 数据收集和管理

数据收集部分提供对数据自动的收集和管理，提供有关所收集数据的标准报告和特别报告。

所收集的数据类型包括：

- 课和课程的概要；
- 测验题目回答；
- 学生成绩数据。

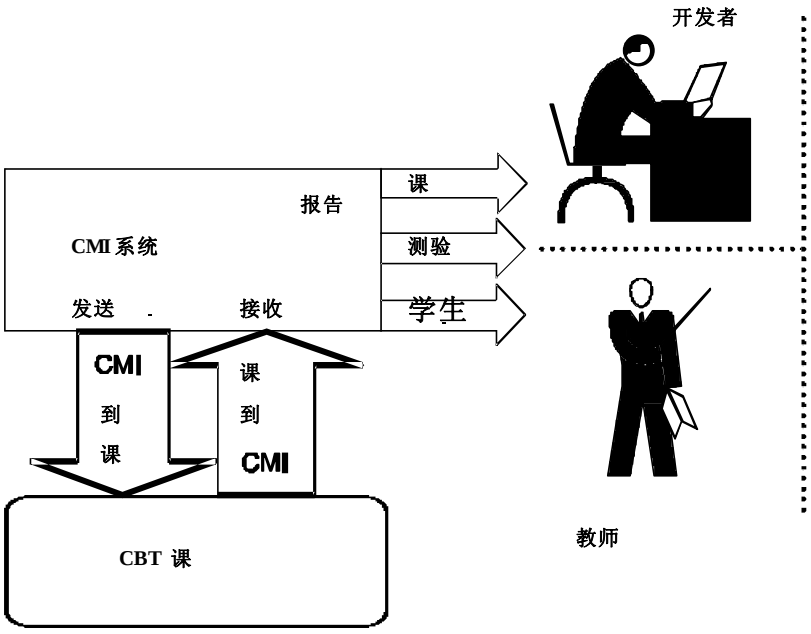

图 4 CMI 系统获得信息的用途

从 CMI系统获得的信息可以是多种类型的，有多种用途：

对于课程开发者

教学开发者使用课程概要数据评估课程，决定怎样改进课程。

对于教师

学生成绩分析。收集学生与教学内容的交互数据，有助于确定学生知道什么，学到了什么。比较同类个体学生的进步可以衡量个人的学习进度。使用学生成绩信息，教师可以为学生掌握学习材料提供指导。

课件分析。决定课程与学习目标是否匹配。精确地确定课件在哪里、为什么没有达到效果。这和题目分析有关。

对于课程开发者和教师

题目分析。这可以表明教学元素教学效果如何，测验题是否能够测定学生表现，从而控制测验和教学的质量。

路径优化。为一个学生定制最合适的教学顺序和测验，决定哪些材料学生可以略过，该生还需要哪些补充材料和补习材料。路径优化与学生表现分析有关。

态度调查。确定学生对课件的喜欢程度，以及学生对课件教学效果的评价。这有助于衡量客户的满意程度。

CMI系统中的数据有两级：

- 第一级是相对精确的课和教学目标数据，布置学习任务安排学习路线需要这些数据；
- 第二级是课程和课程体系分析所需要的数据，如数量众多的测验题信息、CBT 交互、路径数据等。

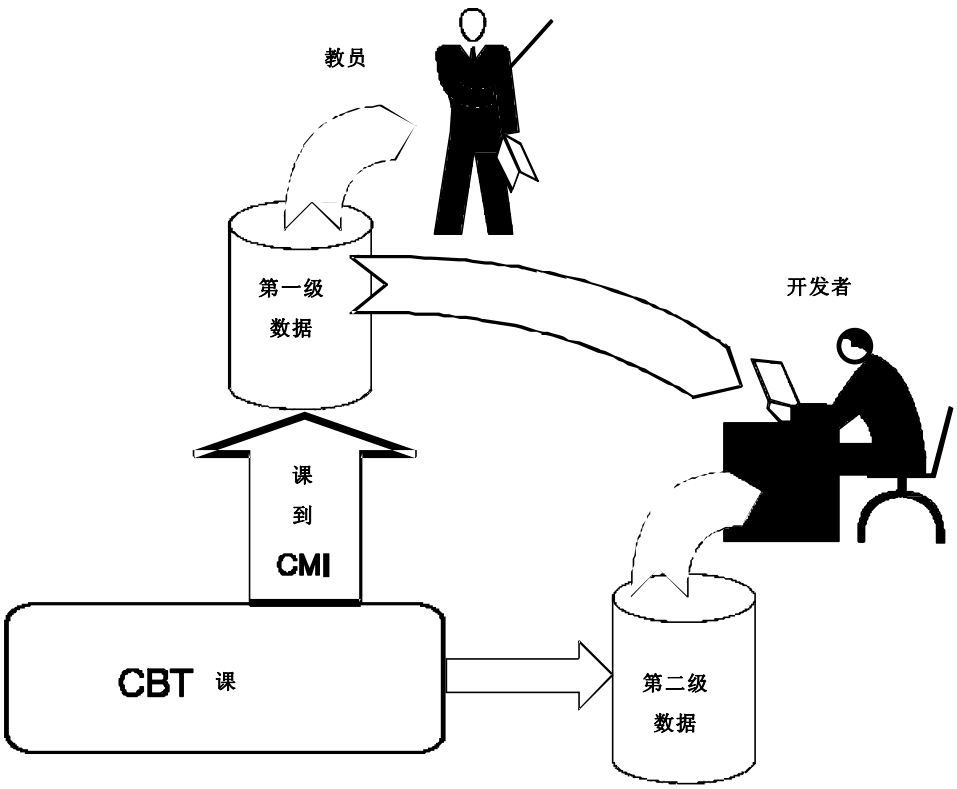

图 5 CMI 系统中的数据级别

不论CMI系统如何实现，都需要两种数据收集系统，这样课件初始化所需要的大量数据不会妨碍布置管理所需要的相对较少的数据。使用不同的数据收集系统，数据可以根据评价和安装需要有选择的开启或关闭。

第一级数据通常总是开启的，但是在有的条件下也会要求关闭它。比如：

- 因为法律原因，不能保存某些行为数据
- 因为管理原因，不需要单个学生的数据.
- 课程只是用来浏览复习的，不需要安排学习路径、纪录行为数据

第二级数据在进行课程小组试用时要对所有课开启。在评估所收集的数据，修改课程中与数据对应的课之后，关闭第二级数据收集。当对某节课进行版本修订时，又会开启数据收集，在小样本范围内评估所收集数据，修改有关的课。

所收集的数据应该能够被标准的统计软件包（如 SPSS 或SAS ）所使用，也应该能够被数据库管理系统(DBMS)所使用。CMI系统最好使用关系型数据库管理系统 (RDBMS) 作为文件管理的基础。

所收集的数据应该能够用于给出关于课件和学生的标准评估报告。使用DBMS 或RDBMS的报表功能能够完成更为复杂的报告。标准报告包括：

- **第一级：**学生表现的历史报告
- **第二级：**课和课程级的分析报告
- **两级：**测验分析报告

可以为授课教师提供他的学生表现如何的标准报告，这个报告可以告知每个学生在每节课上每次努力的情况

用于该报告的数据包括:

- 学生学号或名字
- 课编号
- 课的测验编号(如果有的话)
- 尝试次数
- 完成的数据
- 总尝试时间
- 每次尝试通过/没通过指示
- 每次尝试的分数（如果有的话）
- 未达到的目标

对于课件评估者来说，一种比较好的工具是提供对课程中每节课学习情况的概括报告。从这些数据中可以发现哪节课需要更进一步研究，比如，有高失败率的课，含有高失败率教学目标的课，以及学习时间较长的课。

可用于该报告的数据包括:

a）每节课用时和学生成绩的平均值和标准方差

b）每节课的失败率

1）每次重试的失败率

2）目标失败率

对于课件评估者来说，另外一种较好的工具是关于掌握性测验执行情况的报告。测验题目分析报告能帮助发现不可靠的测验题，它也能够说明一道题的答对与否对于教学目标是否完成或与整个测验是否通过的关系。然而，关于题目与目标、题目与测验的相关性用标准测验试卷的题比用题库的题更能说明问题。

对于该报告有用的数据包括:

a）样本大小
b）采样时间周期 (数据范围)
c）测验失败原因
    1）分数
    2）失败的目标
    3）关键题目或目标
d)目标总结数据(按目标统计)
    1）失败的次数
    2）平均得分
    3）标准方差
    5）每个目标的题目数/通过目标需要完成的题目数
    6）可靠性系数
    7）题目成绩与测验成绩的相关系数
e）题目总结数据 (按项目统计)
    1）难度指数(答对学生的平均数目)
    2）失败次数
    3）题目成绩与测验成绩、目标成绩的关系
    4）题目各选项的分布(每个选项学生选择的次数)
    5）未预料到的回答的统计

第 3 章 互操作性概述

这一章首先介绍本规范所覆盖的 CMI 互操作性的几个方面，说明为什么需要互操作。然后阐述实现互操作所涉及到的数据流，本规范所要定义的数据元素正是这些数据流。

3.1 CMI 的互操作

CMI 系统的互操作体现在：

- 启动课程
- CMI 系统与课之间的通信
- 在 CMI 系统间传递课程结构、学习行为、课程内容
- 存储学生表现数据

3.1.1 启动课程

CMI 将提供每节课启动的标准方法，启动方法要规范化，这样单个 CMI 系统就可以启动来自不同 CBT 供应商的课程。

一般要求

CMI 系统开发者应当建立一种机制启动多种 CBT 运行系统，这个机制可能与 DOS 和 Windows 环境不同，启动机制必须满足下列要求：

- 学生应当单点进出，即学生从 CMI 进入，从 CMI 退出。先进入 CMI，由 CMI 启动 CBT，CBT 运行结束后，将上课结果返回给 CMI。
- 每节课完成时都要把上课结果返回到 CMI 系统。

3.1.2 CMI 通信

通信是使 CMI 和 CBT 系统互操作成为可能的关键。为了实现这个功能，CMI 和 CBT 系统之间必须交换标准类型的数据，这些数据的格式已经预先商议决定。这些数据包括：

- 学生学习经历信息。这些信息会影响学生学习 CBT 课的效率和效果。这种信息在 CBT 课开始时从 CMI 系统获得。
- 在一节 CBT 课中学生的表现信息 (第一级数据)。CMI 需要这些信息以更新学生的记录，调整下面的学习内容安排。
- 项目反应数据，在模拟或练习中的表现，一节课所选定的路径（第二级数据）。这些信息能用于判断每节课的进展情况，精确显示学生的表现。

图 6 描述了 CMI 数据流，“学生表现”数据属于第一级数据，“课记录”数据是第二级数据。

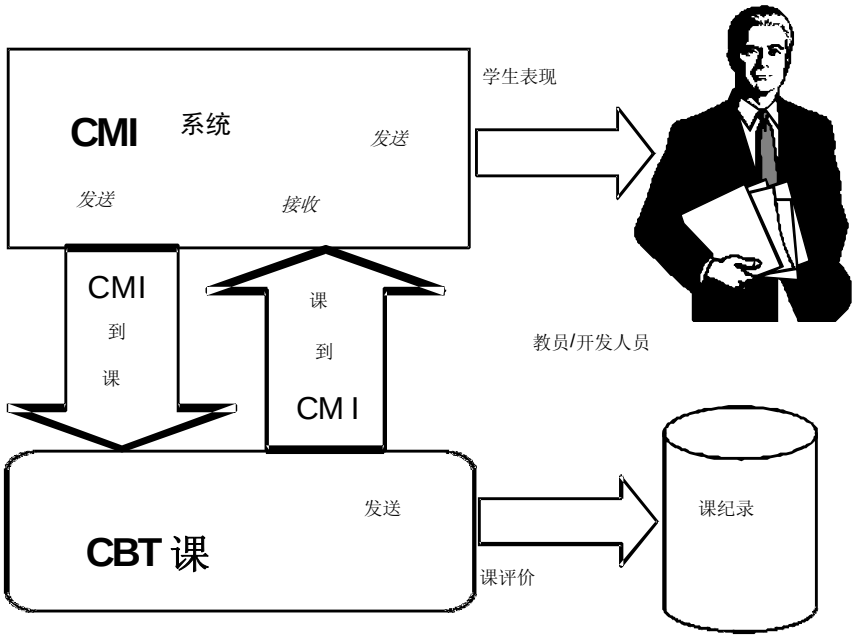

图 6 系统中的数据流

3.1.3 储存课的记录

学生表现数据应当包括由 CBT 的课或测验产生的，用于 CMI 系统或其他分析工具的信息。还包括课和学生的数据，这些数据是冗余的，在多个 CMI 系统的情况下需要用到这些信息。

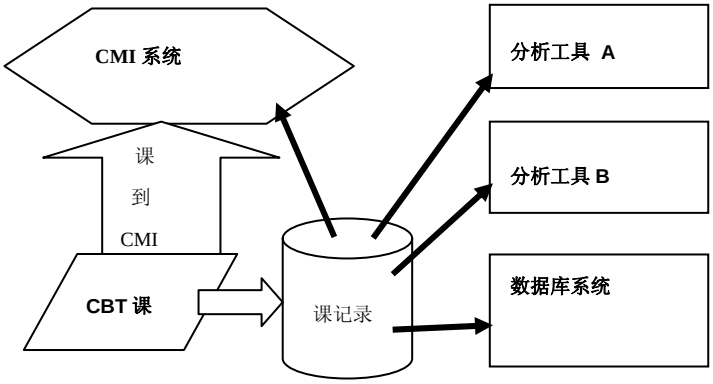

图 7 详细的学生记录

标准化学生表现记录的格式就可以让更多的工具使用这些信息：

CMI. 有些 CMI 系统能够分析或使用比标准“课-CMI”通信更多的信息。定义一个标准化格式存储额外的学生表现数据，这些 CMI 系统就能够访问所有的数据。

分析工具. 有多种分析工具可以用来分析学生表现和课程的运行数据。标准化的存储格式使得这些工具能够分析来自不同课的信息。

竞争力. 有了标准交换格式，分析工具的市场会扩大很多，不再局限于一家卖主的顾客。这样由于可能的更大的消费群体，生产者就会出于经济利益上的激励而努力生产更成熟

的、容易使用的分析工具。

3.1.4 移动课程

一门课程可以很简单，只有几节课，顺序阅读；也可以很复杂，有好几百节课，并且其中有的课之间有先学后学关系，有的课则没有什么学习次序要求。课程应该有三个组成部分：

- 教学要素
- 结构
- 行为要素

教学要素应当包括课程中所有的课、测验和其他可分配单元。这些要素通常被称为内容。内容的描述可能包括所有在这个课程中需要掌握的目标。

结构决定了对于每个学生应当按什么次序学习。这种次序可以很简单，只是课的本身的次序，也可以很复杂，要依据先修条件或学生的表现决定。课序编列作为 CMI 系统的一部分对课程内容进行排序。

行为可以分为两类：一节课中的行进逻辑和课与课之间的行进逻辑。课程行为主要与后者有关，而前者，即在一节课内的行为，可以基于 CMI 传给课的信息。

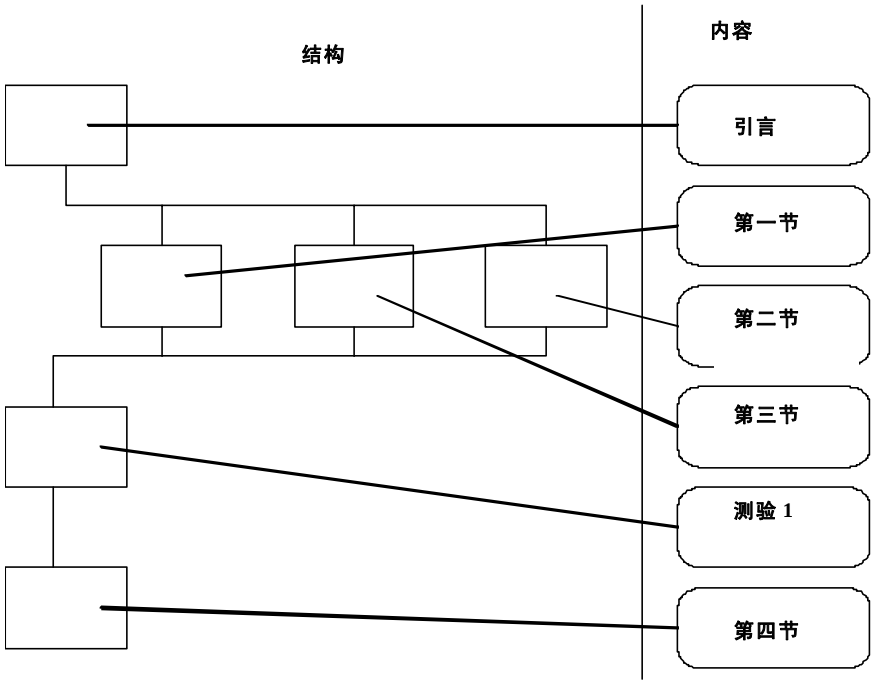

图 8 课程结构

3.2 互操作性的关键：通信

互操作性的实现需要包括以下数据流：

- CMI ⇔ CBT
- CMI ⇒ CMI
- CBT ⇒课分析工具

一个完整的互操作性实现至少包括 CMI/CBT 和 CMI/CMI 的数据流。这一部分描述最基本的数据流。

3.2.1 CMI 和 CBT 之间的数据流

CMI 和课之间的通信有两种方式。当每节课开始时，CMI 系统将数据发送到课；当每节课结束时，课再把信息发送回 CMI 系统。

图 9 示意了 CMI 和课件系统之间的数据流。在这个数据流里有两类数据：

- 在每节课开始时由 CMI 产生，CBT 课接收的数据。
- 由 CBT 课传送到 CMI 的数据。它使 CMI 系统可以纪录学生的表现，决定下节课的内容安排

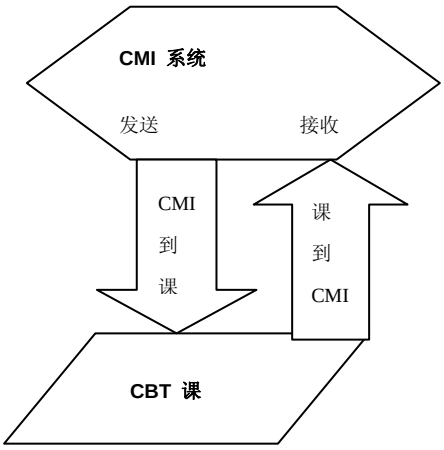

图 9 CMI 和 CBT 之间的数据流

3.2.2 CMI 系统之间的数据流

图 10 添加了一个信息通道，在不同的 CMI 系统之间要传送课程结构和学生学习路径等数据。课程结构应当对课程有足够详尽的描述，使 CMI 系统能够理解其结构、内容和行为。

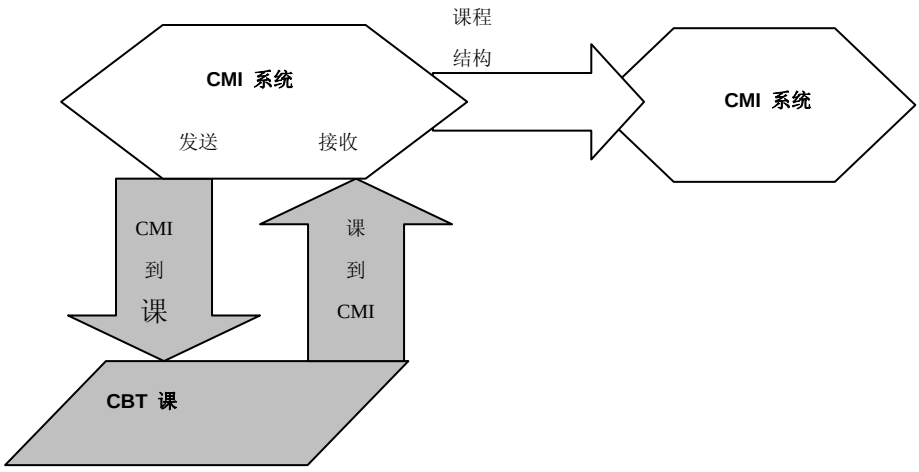

图 10 添加课程结构的数据流

3.2.3 从 CBT 到分析工具的数据流

图 11 描述了从每节课获得的课程数据，可以为多种用于题目分析和课程评价的工具所使用。

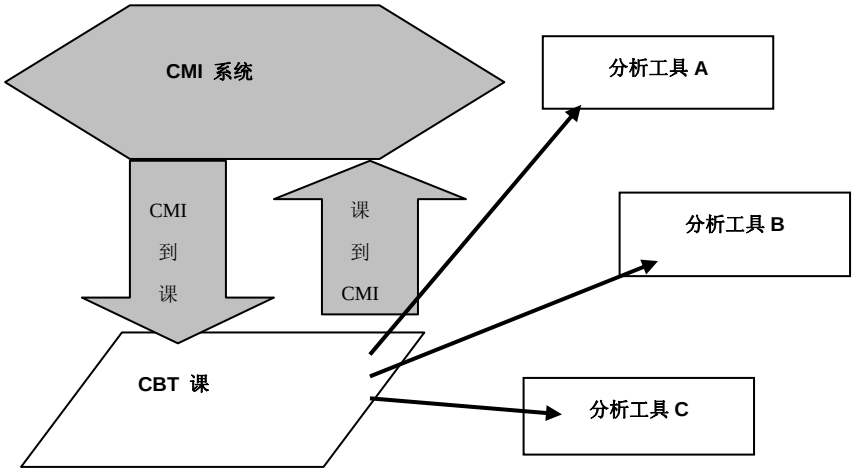

图 11 学生记录数据流

数据也可以通过 CMI （图 12）流向分析工具。通常 CMI 系统都具有一些数据分析能力，如果需要进行更进一步的分析，则数据可被传送到有关的工具。当在 web 上传送学习内容时，因为课不具备向分析工具发送信息的能力，所以必须将信息送到 CMI 系统，由 CMI 作进一步发送。

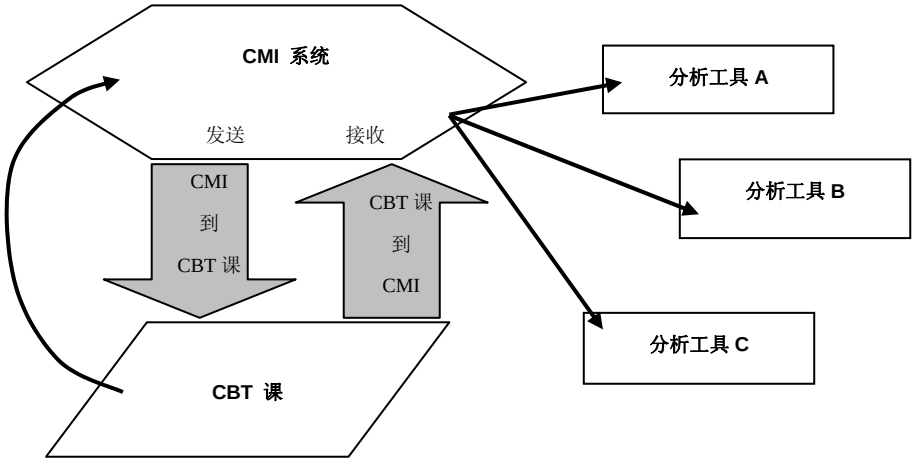

图 12 学生记录可选择的数据流

第 4 章 数据结构概述

4.1 基本数据结构

本规范中结构化数据使用层次模型。数据元素中可以含有数据元素。在顶级，数据元素通常叫做一个类。为了清楚表明每种数据元素在这个层次框架中的位置，在顶级之下的每级名称前都用一个竖线，两个破折号表示。表 1 为这种层次表示法的示例。

表 1 层次结构的可视化表示

层次级	表格描述
类别-层次中的顶级	数据元素
层次中的第二级	|--数据元素
第二级的下一级	|--|--数据元素
第四级的父级	|--|--|--数据元素
层次的第四级	|--|--|--|--数据元素

4.2 数据类型定义

这里定义的数据类型用来描述数据元素的格式。所有的数据类型都以“CMI”开头，以表明他们是 CMI 数据模式所特有的。所有的数据类型都使用 ASCII 编码。

1. CMIDecimal（标准实型变量）

一个带符号的允许小数点的实型数，在前面没有“－”号时为正。示例：“2”、“－2”和“2.2”

2. CMIIdentifier（标准代号）

不包括空格、逗号和非可视字符的数字-字符串，最大长度 255。

3. CMISIdentifier（标准简单代号）

一组数字－字符串，由简单字符开头（A、B 或 J）且以整数结尾。尾随这个字符可以是 1 到 5 个数字。字符在 HTTP 通讯中大小写均可，在 API 通讯中只能是小写。

4. CMIString255（标准短字符串）

由 ASCII 码组成的长度小于 255 的字符序列。

5. CMIString4096（标准长字符串）

由 ASCII 码组成的长度小于 4096 的字符序列。

6. CMITimespan（标准时间间隔）

由小时、分和秒表示的时间段。形如 HHHH:MM:SS.SS，中间用冒号隔开。小时由两到四位数字组成。分由两位数字组成，可以是 00 到 59 之间的任意值。秒由两位数字和额外的两个小数位组成，可以是 00 到 59.99 之间的任意值。三个由冒号分隔的数字都是需要的，只能显示分和秒的情况也不例外。

7. CMIVocabulary（标准关键字）

表 2 里列举的是计算机教学管理系统（CMI）数据结构中使用的关键字：

表 2 CMI 数据结构中使用的关键字

关键字种类	关键字	
Mode（模式）	normal（普通）	review（复习）
	browse（浏览）	
Status（状态）	passed（通过）	completed（完成）
	failed（失败）	incomplete（未完成）
	browsed（已浏览）	not attempted（未尝试）
Exit（退出）	time-out（超时）	suspend（等待）
	logout（注销）	
Why-left（退出原因）	student selected （学生选择）	lesson directed （课程指示）
	exit（退出）	directed departure （指示的离开）
Credit（学分）	credit（有学分）	no credit（无学分）
Entry（入口）	ab-initio（初次）	resume（继续）
Time Limit Action （限时反应）	exit, message （提示后离开）	exit, no message （不提示离开）
	continue, message （提示后继续）	continue, no message （不提示继续）
Interaction（交互）	true-false（是/否）	choice（选择）
	fill-in（填充）	matching（配对）
	performance（表现）	likert（单个字符）
	sequencing（顺序）	Unique（唯一）
	numeric（数字）	
Result（评判）	correct（正确）	wrong（错误）
	unanticipated （未预计到的结论）	neutral （中性结论）
	x.x (CMIDecimal) （给分）	

第 5 章 CMI 与 CBT 通信信息规范

本章中我们将“规范”所定义的 CMI 与 CBT 间交换的标准数据元素以列表形式罗列出来，以方便读者对照理解后面的应用实例。具体的数据元素定义请参见所发布的规范草案第 5、6 章。

5.1 表格说明

下面介绍表格中每列标题的含义：

名称

数据元素名。它可能与具体实现所使用的字段名称不一样。

含义

数据元素的语义。同名元素在不同环境下含义不同。例如每节课有“分数”，该节课的某个目标也有“分数”。这应该是不同的“分数”。 数据元素的含义是根据它出现在哪个层次决定的。

表值

这列标明数据元素或数据类是只有一个值（S），还是可以有多个值。如果它可以取多值，还要进一步标明这些值是有序的（+），还是无序的（*）

数据类型

数据元素的值必须采用本列定义的格式，或者为一个字符串，该字符串可以被转换为所定义的格式。

5.2 CMI 系统到 CBT 课

这些信息是一节课为了能够完成它预期的功能，从CMI系统获得的信息。在这“规范”里，首先列出核心词，继之以按字母顺序排列的可选类别名。每一类别中给出了适合于该类别的数据元素。

表 3 CMI 系统到 CBT 课的数据元素

Name	中文名称	含义	表值	数据类型
Core	核心数据	要求所有的 CMI 系统都提供的信息	S	-
|--Student ID	学生学号	CMI 系统每个用户的唯一标识，字母-数字代码/标识符	S	CMIIdentifier
|--Student Name	学生姓名	学生在课程登记时使用的正式名字。应该是完整的姓名，而不仅仅是名	S	CMIString255
|--Output Mechanism	输出机制	确定可分配单元在它结束的时候如何与 CMI 系统交换信息	S	CMIString255
|--Lesson Location	课的位置	退出时传递给 CMI 系统的退出点	S	CMIString255
|--Credit	学分	表明学生在本节课的学习（通过/失败和分数）是否得到 CMI 系统的学分	S	CMIVocabulary
|--Lesson Status	课的状态	由 CMI 系统决定的当前学生状态，当课启动时，这个状态数据要传递给课	S	CMIVocabulary
|--Entry	入口	表明学生以前是否学习过此课	S	CMIVocabulary
|--Information Store	信息存储	向 CBT 课指出关于学生进度和课程状态的信息是如何存储的	S	CMIString255
|--Score	分数	学生在本节课上次学习中的表现	S	--
|--|--Raw	原始得分	用数值形式表示的学生在课中的成绩。可能是没有处理的原始分数	S	CMIDecimal
|--|--Maximum	最高分	学生取得的最高分数或总分数	S	CMIDecimal
|--|--Minimum	最低分	学生取得的最低分数		CMIDecimal
|--Total Time	总时间	学生学习这节课的累计用时	S	CMITimespan
|--Lesson Mode	课的模式	与学生相关的信息，可用来改变课的行为	S	CMIVocabulary
Suspend Data	暂停数据	前次使用课时产生的信息，希望将此数据存储为将来所用	S	CMIString4096
Launch Data	启动信息	在课创建时产生的信息，每次课启动时都需要	S	CMIString4096
Comments	评语	教师针对学生的评语，在适当的时候由课呈现给学生	S	CMIString4096
Evaluation	评估	可分配单元可以产生详细的学生表现/课程评估信息。这个类别说明怎样收集这些信息以及在哪儿存放这些信息	S	-
|--Course ID	课程编号	字母数字序列，每门课程的唯一标识	S	CMIIdentifier
|--Comments	评语	确定怎样收集学生对一节课的评语	S	CMIString255

Name	中文名称	含义	表值	数据类型
|--Interactions	交互	确定将怎样收集一节课中学生交互的详细记录	S	CMIString255
|--Objectives Status	目标状态	确定将怎样收集一节课中各个目标的完成状态	S	CMIString255
|--Path	路径	确定在学生学习一节课的过程中怎样收集其学习路径的详细信息	S	CMIString255
|--Performance	表现	确定在一个复杂特定的情形中（比如模拟）如何收集有关学生表现的详细信息。	S	CMIString255
Objectives	教学目标	学生完成本节课每个教学目标的情况	*	-
|--Identifier	目标编号	开发者定义的，特定课的目标编号	S	CMIIdentifier
|--Attempts	尝试次数	当前可分配单元某目标的学习次数	+	--
|--|--Score	目标得分	为掌握该目标学生每次学习得到的分数	S	--
|-|-|-Raw	原始分数	用数值形式表示的学生在课中的成绩。可能是没有处理的原始分数	S	CMIDecimal
|--|--|--Maximum	最高分数	学生在此目标上测验的最高得分	S	CMIDecimal
|--|--|--Minimum	最低分数	学生在此目标上测验的最低得分	S	CMIDecimal
|--|--Status	目标状态	为掌握该目标学生每次学习后的状态	S	CMIVocabulary
Student Data	学生数据	基于学生表现定制一节课所需要的信息	S	-
|--Attempt Number	尝试次数	学生上这节课的次数	S	CMIInteger
|--Mastery Score	及格分数	在课堂之外事先规定的及格分数。	S	CMIInteger
|--Max Time Allowed	最大允许时间	允许学生在当前课中学习的时间	S	CMITimespan
|--Time Limit Action	限时反应	当超过最大允许时间时，课要采取的行动	S	CMIVocabulary
|--Attempt Records	尝试纪录	学生前几次在上这课的表现	+	-
|--|--Lesson Score	课得分	学生每次上这节课的得分	S	CMIInteger
|--|--|--Raw	原始分数	用数值形式表示的学生每次尝试的成绩。可能是没有处理的原始分数	S	CMIDecimal
|--|--|--Maximum	最高分数	学生取得的最高分数或总分数	S	CMIDecimal
|--|--|--Minimum	最低分数	学生取得的最低分数	S	CMIDecimal

Name	中文名称	含义	表值	数据类型
|--|--Lesson Status	课状态	表示学生每次尝试后课的状态	S	CMIVocabulary
Student Demographics	学生个人信息	学生进入课程前就具有的属性	S	-
|--City	城市	学生当前地址的一部分	S	CMIString255
|--Class	班级	学生隶属的一个预先确定好的培训组织	S	CMIString255
|--Company	公司	学生被雇用的单位	S	CMIString255
|--Country	国家	学生当前地址的一部分	S	CMIString255
|--Experience	经历	有关学生过去学习情况的一些信息，课需要这些信息决定教学内容和教学策略	S	CMIString255
|--Familiar Name	昵称	用来指代学生的非正式名称	S	CMIString255
|--Instructor Name	教师姓名	负责帮助学生理解课内容的人的姓名	S	CMIString255
|--Title	头衔	当前学生拥有的学位或职位的名称	S	CMIString255
|--Native Language	母语	学生出生国家所使用的语言	S	CMIString255
|--State	省份	包括直辖市，自治区等	S	CMIString255
|--Street Address	街道地址	学生当前地址的一部分	S	CMIString255
|--Telephone	电话	学生的电话号码	S	CMIString255
|--Years Experience	工作年限	学生处于当前或类似职位的年限	S	CMIString255
Student Preference	学生偏好	参数选择，会影响后续课	S	-
|--Audio	音频	声音开/关，音量控制	S	CMISInteger
|--Language	语言	信息发送所用的语言	S	CMIString255
|--Lesson Type	课类型	课参数选择对当前课的的适用程度	S	CMIIdentifier
|--Speed	速度	内容传递步调	S	CMISInteger
|--Text	文本	文本可视化控制	S	CMISInteger
|--Text Color	文本颜色	文本的前景色和背景色	S	CMIString255
|--Text Location	文本位置	文本窗口在屏幕上的位置	S	CMIString255
|--Text Size	字号	屏幕上文本字符尺寸	S	CMIString255
|--Video	视频	屏幕上运动图像的色调和亮度	S	CMIString255
|--Windows	窗口	视频、帮助和词汇等窗口的大小和位置	*	CMIString255

5.3 CBT 课到 CMI 系统

这里给出的是一节课需要向 CMI 系统提供的信息。首先是核心项，为每节课必须提供的信息，其次是按字母顺序排列的可选项。

表 4 CBT 课到 CMI 系统数据元素

名称	中文名称	含义	表值	数据类型
Core	核心数据	CMI 系统运作所需要的信息。	S	-
|--Lesson Location	课的位置	标记学习结束时离开课的位置。	S	CMIString255
|--Lesson Status	课状态	此次学习结束时学生的状态。	S	CMIVocabulary
|--Exit	退出情况	标明学生如何或为何离开该节课。	S	CMIVocabulary
|--Score	成绩	学生这次学习此节课的最终成绩。	S	--
|--|--Raw	原始得分	学生在这节课中的表现 ，数值表示，可能是未经处理的成绩	S	CMIDecimal
|--|--Maximum	最高分	学生在这节课中的最高成绩	S	CMIDecimal
|--|--Minimum	最低分	学生在这节课中的最低成绩	S	CMIDecimal
|--Session Time	学习用时	此段学习的时间	S	CMITimespan
Suspend Data	暂停数据	由课产生的、将来再学这节课时所需要的信息，这些信息要传给 CMI 系统，在下次学生又进入这节课时由 CMI 返回	S	CMIString4096
Comments	注释	本次上课学生时所做笔记或记号	S	CMIString4096
Objectives	教学目标	学生完成本节课每个教学目标的情况	*	--
|--Identifier	目标编号	开发者定义的，特定于课的教学目标编号	S	CMIIdentifier
|--Attempts	尝试次数	当前可分配单元某目标的学习次数	+	--
|--|--Score	目标得分	为掌握该目标学生每次学习得到的分数	S	--
|--|--|--Raw	原始分数	学生在此目标上每次测验的分数值，可能是没有处理的原数据	S	CMIDecimal
|--|--|--Maximum	最高分数	学生在此目标上测验的最高得分	S	CMIDecimal
|--|--|--Minimum	最低分数	学生在此目标上测验的最低得分	S	CMIDecimal
|--|--Status	目标状态	为掌握该目标学生每次学习后的状态	+	CMIVocabulary

名称	中文名称	含义	表值	数据类型
Student Data	学生数据	在不离开课的情况下，对于该节课某部分内容，学生每次学习的情况	S	-
|--Tries During Lesson	学习次数	学习该节课或选定内容的总次数	S	CMIInteger
|--tries	学习纪录	与每次学习相关的数据	*	
|--|--Score	学习成绩	每次学习后的成绩	S	--
|--|--|--Raw	原始分数	学生在此节课每次学习的分数值，可能是没有处理的原数据	S	CMIDecimal
|--|--|--Maximum	最高分数	学生测验的最高得分	S	CMIDecimal
|--|--|--Minimum	最低分数	学生测验的最低得分	S	CMIDecimal
|--|--Status	状态	每次学习后课的状态	S	CMIVocabulary
|--|-- Time	用时	每次学习所用时间	S	CMITimespan
Student Preferences	学生偏好	参数选择，会影响后续课	S	-
|--Audio	音频	声音开/关和音量控制	S	CMISInteger
|--Language	语言	信息发送所用的语言	S	CMIString255
|--Lesson Type	课类型	课参数选择对当前课的的适用程度	S	CMIIdentifier
|--Speed	速度	内容传递步调	S	CMISInteger
|--Text	文本	文本可视化控制	S	CMISInteger
|--Text Color	文本颜色	文本的前景色和背景色	S	CMIString255
|--Text Location	文本位置	文本窗口在屏幕上的位置	S	CMIString255
|--Text Size	字号	屏幕中文本字符尺寸。	S	CMIString255
|--Video	视频	屏幕上运动图像的色调和亮度	S	CMIString255
|--Windows	窗口	视频、帮助和词汇等窗口的大小和位置	*	CMIString255

5.4 CMI 与 CBT 之间传递数据比较

为了让读者更清晰的了解 CMI 系统到 CBT 课，CBT 课到 CMI 系统数据传递的相同与不同，我们在表 5 中对这两种数据流进行比较。

表 5 CMI 系统与 CBT 课之间交换数据比较

从 CMI 到 CBT 课数据流	从 CBT 课到 CMI 系统数据流
CORE（核心数据） *Student ID（学生学号）* *Student Name（学生姓名）* *Output Mechanism（输出机制）* *Lesson Location（课的位置）* *Credit（学分）* *Lesson Status（课的状态）* *Entry（入口）* *Information Store（信息存储）* *Score（分数）* *Total Time（总时间）* *Lesson Mode（课的模式）*	CORE（核心数据） *Lesson Location（课的位置）* *Lesson Status（退出状态）* *Exit(退出情况)* *Score(成绩)* *Session Time(学习用时)*
SUSPEND DATA（暂停数据）	SUSPEND DATA（暂停数据）
LAUNCH DATA（启动信息）	
COMMENTS（评语）	COMMENTS（注释）
EVALUATION(评估) *Course ID（课程编号）* *Comments（注释）* *Interactions（交互）* *Objectives status（目标状态）* *Path（路径）* *Performance（成绩）*	
OBJECTIVES（目标） *Identifier（目标编号）* *Attempts（尝试次数）*	OBJECTIVES（教学目标） *Identifier(目标编号)* *Attempts（尝试次数）* *Score（成绩）* *Status（目标状态）*
STUDENT DATA（学生数据） *Attempt Number（尝试次数）* *Mastery Score(掌握得分)* *Max Time Allowed（最大允许时间）* *Time Limit Action（限时反应）* *Attempt Records（尝试记录）*	STUDENT DATA（学生数据） *Tries During Lesson（学习次数）* *Tries（学习纪录）*
STUDENT DEMOGRAPHICS（学生个人信息） *City（城市）* *Class（班级）* *Company（公司）* *Country（国家）* *Experience(经历)* *Familiar Name(昵称)* *Instructor Name（教师姓名）*	

Title（头衔） *Native Language（母语）* *State(省份)* *Street Address（街道地址）* *Telephone(电话)* *Years Experience(工作年限)*	
STUDENT PREFERENCE（学生偏好） *Audio（音频）* *Language（语言）* *Lesson Type（课类型）* *Speed（速度）* *Text（文本）* *Text Color（文本颜色）* *Text Location（文本位置）* *Text Size（字号）* *Video（视频）* *Windows（窗口）*	STUDENT PREFERENCES（学生偏好） *Audio（音频）* *Language（语言）* *Lesson Type（课类型）* *Speed（速度）* *Text（文本）* *Text Color（文本颜色）* *Text Location（文本位置）* *Text Size（字号）* *Video（视频）* *Window（窗口）*

第 6 章 应用实例——CMI/CBT 通信的 HTTP 绑定

在本章中我们为大家列举一个在 HTTP 环境下实现 CMI 系统与 CBT 课之间通信的应用实例。

6.1 HTTP 下 CMI/CBT 通信概述

HTTP 是一个客户机/服务器协议。一个客户机程序（通常是一个浏览器）发出请求，一个服务器程序（万维网服务器）响应请求。通过 HTTP 协议，在不同的地理位置，客户机和服务器程序可以运行在相同类型或不同类型的计算机上。 CMI 的一些部分作为万维网服务器的一部分运行，而另一些部分（学生用户界面）则作为万维网浏览器运行。

这一节描述了如何在因特网上实现数据流动。数据必须转换成一种 ASCII-文本格式的组/关键字数据，即被划分为若干组的 ASCII 流，且大部分组都包含关键字。

6.1.1 CMI 与 CBT 通信的数据流

如图 13 所示，一个符合要求的实现必须包含以下的数据流：

- CMI 到 CBT：这个流由 CMI 系统在启动 CBT 课程的时候建立，然后由 CBT 课读取以获取它所需要的学生信息。
- CBT 到 CMI：这个流在学生完成学习时由 CBT 课建立，包含了 CMI 系统所需要的数据。

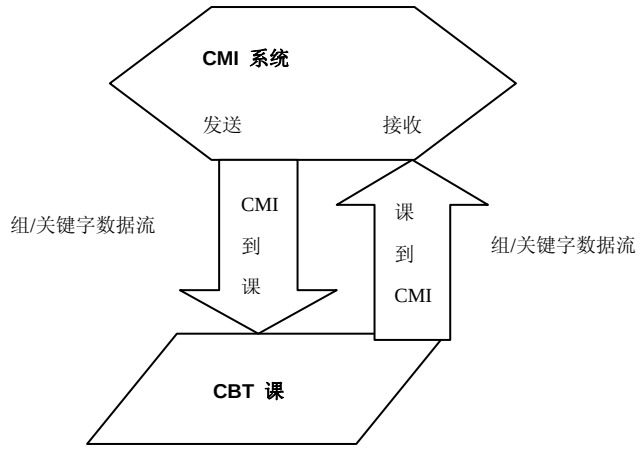

图 13 CMI/CBT 通信

数据大体包括以下内容和格式。

CMI 到 CBT 数据：

- 内容：课发挥最佳功能所需要的信息。
- 数据格式：组/关键字

CBT 到 CMI 数据

- 内容：CMI 系统跟踪学生的绩效，制定新的教学安排所需要的信息，。
- 数据格式：组/关键字

6.1.2 通信数据协议：HTTP

本节参考 HTTP 标准，并且描述实现 CMI 所必须的 HTTP 协议命令。

HTTP 1.0 规范 在 RFC 1945（网络标准草案）中进行了描述。

其它相关标准草案（RFC）包括：

- 用于 ARPA 因特网文本消息格式的 RFC 822 标准 。
- RFC 1738 统一资源定位器(URL)
- RFC 1808 相对统一资源定位器
- RFC 1521 MIME (多用途的网际邮件扩充协议) 第一部分：详细说明和描述因特网消息体格式的机制。

6.1.3 通信数据格式

在 CMI 系统和 CBT 课之间通讯的数据的格式必须是表 6 所示的组/关键字数据流。这个格式支持三种类型的数据：组、关键字和注释。

表 6 通讯格式

表项	元素名称
[组]<cr>	组
关键字=参数<cr>	关键字
;组和关键字<cr> ; 可能含有注释<cr>	注释

6.1.3.1 组

组提供了一种机制可以把一个文件切分为多个便于管理的段，这样使恢复数据变得更加简单。组还提供了将文件组织成彼此具有逻辑关系的部分的办法。无论是人还是电脑处理文件，组都是有极大帮助的。

组是逻辑上相关联的数据元素的集合，可以多于一行。组包括从一个组标志符到另一个组标志符之间的内容。尽管组可以包含关键字，但它们不能包含其它的组。

组的名称必须由空格或缩进(tab)开始，并被包括在方括号“[]”内，它的内容可以包括

数字和字符或者下划线元素。除开空格和 Tab 的其它任何字符都不允许出现在左括号之前。组的名称要区分大小写。

在组关键字之间的所有符号和回车的意义都由组的具体定义决定。如果一个组包含关键字，那么空行和多余的回车都将被忽略。

组可以以任何顺序出现，但只当第一次出现时才有意义。

组名称示范：

[comments]　　这是一个名为“注释”的组

[OBJECTIVES_STATUS]　　这是一个名为“目标状态”的组

6.1.3.2 关键字

关键字是数据项的名称，它们被限制在一行以内。包含标点的数据项的总长度不能超过 255 个字符。与关键字关联的数据项指的是关键字参数或关键字值。

关键字必须是左对齐的，后面跟着等号（＝），它由数字和字符或下划线元素组成。在等号之前和之后的空格将被忽略。对于关键字将区分大小写。在各关键字之间的空行将被忽略。

关键字必须是组的成员，但组不一定要有关键字。**[COMMENT]**（注释）组是一个没有关键字的组的例子。

单个组中的关键字必须不同。如果关键字重复，那么只有第一个关键字被识别。重叠的关键字可以通过在后面增加点号和 0 到 9999 之间的整数来进行区分。由此产生的顺序关系并不重要。

不同组的关键字可以重复。举例来说，同一个文件可以在**[LESSON_DATA]**组中和**[STUDENT_DATA]**组中同时包含关键字 **ID**，当然这些关键字是不同的。

6.1.3.3 注释

注释是用来方便用户阅读文件的。注释没有具体意义，也不会被处理。由分号开头代表此行是注释。在分号之后和回车之前的所有内容都被视为注释内容。注释不与组名和关键字放置在同一行上，举例如下。

```
[CORE]
LESSON_STATUS = Passed
LESSON_LOCATION = End
SCORE = 87
TIME = 00:25:30
; 这是数据的核心组
; 这是已通过的课程表现数据，耗时 25 分 30 秒
; 得 87 分
```

6.1.3.4 示例

[Objectives_Status]组是一个包含多个关键字的组。它有多个对象 ID，每个 ID 包括多个不同的状态。

```
[Objective_Status]
J_ID.1= AB112
J_Status.1 = Pass
J_ID.2= AB124
J_Status.2 = Pass
J_ID.3= AB196
J_Status.3 = Fail
```

在等号之后的第一个非空格字符指示了数据的开始。关键字中的大小写和空格都是可识别的，这取决于与关键字相连的数据的定义。在下面的例子中，输出文件是区分大小写的，而且学生 ID 可能需要大写。

```
Student_ID = JQH2142
OUTPUT_FILE=C:\STURECS\JQH2142.DTA
postal_code = 98124-2207
```

6.2 开始 CBT 课

概括的说，基于万维网的 CBT 启动顺序如下：

- 学生从 CMI 的用户界面（菜单）选择一个 CBT 可分配单元（AU）。
- CMI 将一个 URI (统一资源识别器) 位置以及启动参数送给 HTTP 客户机。
- HTTP 客户机向 HTTP 服务器请求 CBT 可分配单元内容。
- HTTP 服务器复制程序和数据到 HTTP 客户机。
- CBT 可分配单元在启动前从 HTTP 客户机取得 URI 参数，并与 CMI 系统开始会晤。

图 14 显示的是一个典型的启动顺序：

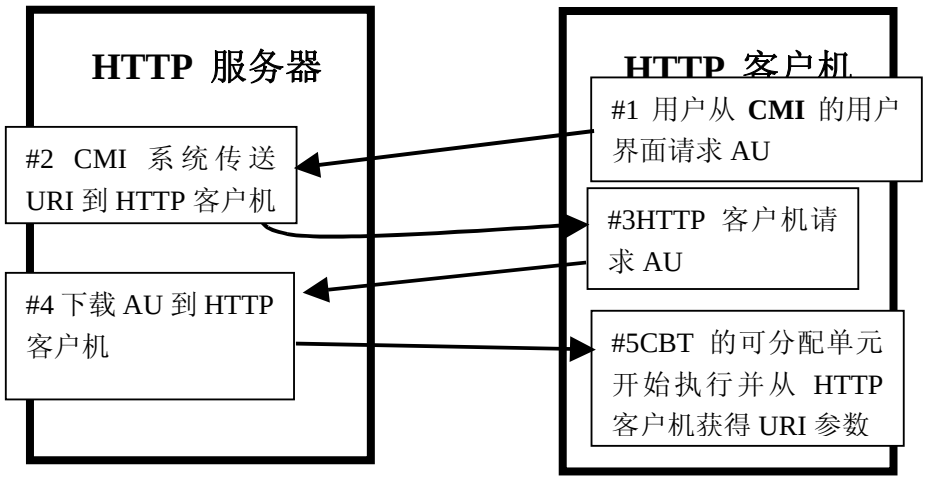

图 14 基于万维网启动顺序

6.2.1 CBT AU/URL 命令行

当 CMI 系统启动一个 CBT 可分配单元（AU），它给 HTTP 客户机程序（如万维网浏览器）发送一个统一资源定位器（URL）命令行。这个命令行包含三个基本含义：

- CBT 可分配单元的统一资源定位器（URL）
- IEEE 需要的参数
- 万维网启动参数

6.2.1.1 CBT 可分配单元的 URL

CBT 可分配单元的统一资源定位器（ URL）是可分配单元在万维网上全球性的唯一的地址。这个地址的第一部分表示了它使用了哪种协议。

使用规则：

- URL 开头应是“http://”。
- URL 中间不包含问号（?）。

示例：

下面两个 URL 指向域名为 sandybay.co 中的两个不同的文件。

http://www.sandybay.com/cgi-bin/lesson.cgi /param1

http://www.sandybay.com/index.html

6.2.1.2 IEEE 需要参数

表 7 给出了将使用到的参数

表 7 必须的 IEEE 参数

IEEE 参数名	**格式**	**值**
AICC_SID	字母-数字的 示例： AICC_SID=1234WE9	学习段 ID 号。这是个用来唯一标识可分配单元学习段的字符串，用以与其他 CMI/CBT 正在活动的学习段区别。 CMI 系统在可分配单元启动时生成并传递该值。 可分配单元向 CMI 系统做出请求时，使用该值作为标识。
AICC_URL	统一资源定位器（URL）位置 示例： AICC_URL=http%3A%2F%2F company.com%2Fcgi-bin%2F CMI.cgi	完整的 URL，包括可分配单元向 CMI 系统发送请求使用的协议。 这可能类似一个服务器上的公共网关接口（CGI）服务程序。这个程序对于 CMI 系统类似 CGI 的“捕捉器”。

示例：

URL 编码
AICC_sid=123&AICC_URL=http%3A%2F%2Fcmi.net%2FCGIBin%2FCMI.cgi&vendorparam=this

非 URL 编码
AICC_sid=123&AICC_URL=http://cmi.net/CGI-Bin/CMI.cgi&vendorparam=this

6.2.1.3 万维网启动参数

万维网启动参数是课特有的。它们是添加到课文件名中的字符串，用于在万维网环境中成功地启动一个可分配单元 。 这些参数是独立的可分配单元所要求的附加参数。

6.2.1.4 启动行格式

启动行具有这样的形式：

<CBT 可分配单元的 URL >?<IEEE-需要参数 1>&<IEEE-需要参数 2>& <万维网启动参数>

此处：

- <CBT 可分配单元的 URL > 是万维网启动所需要的可分配单元原始文件的 HTTP 位置(注意不能有问号，问号只能用做 IEEE 命令行隔离符)。
- <IEEE-需要参数 1> 代表被启动的可分配单元所需要的参数 AICC_SID 或 AICC_URL。参数需求和规则在 6.6 节中描述。
- <IEEE-需要参数 2> 代表 AICC_SID 或 AICC_URL，但它们都不是参数 1。
- <万维网启动参数> 是课的特性参数，它们包含在课程互换可分配单元文件的 WEB_LAUNCH 字段中。

命令行是由包含在可分配单元文件中的两个字段——**File_Name** 和 **Web_Launch**，以及 CMI 生成的所需参数串连而成的。文件名由带有问号的参数分开。每一个参数字段由“&”分开。所有的值必须是 URL 编码的。

数据格式

数字和文字。 值区分大小写。问号右边的所有字母限制在 255 个字符内。

6.3 CMT/CBT 在学习段中的通信

CBT 和 CMI 具有客户机/服务器的关系。在这里，CMI 是服务器，CBT 可分配单元是客户机。CBT 对 CMI 的每个请求是通过 POST 方法访问公共网关接口（CGI）的 URL 实现的。CGI 的 URL 在 URI 启动参数中有所说明。

下面是一个典型 CMI 和 CBT 可分配单元的在学习段中的通信过程：

- 可分配单元（AU）产生一个独立的 HTTP 学习段
- AU 发送一个信息请求 CMI 系统发送的启动课程信息
- 在 AU 学习段结束之前，AU 向 CMI 系统发送学生绩效和课的状态数据

- 当学生退出 AU，AU 发送一个 “end session（结束学习段）”消息给 CMI 系统。

图 15 展示了典型的学习段中的通信：

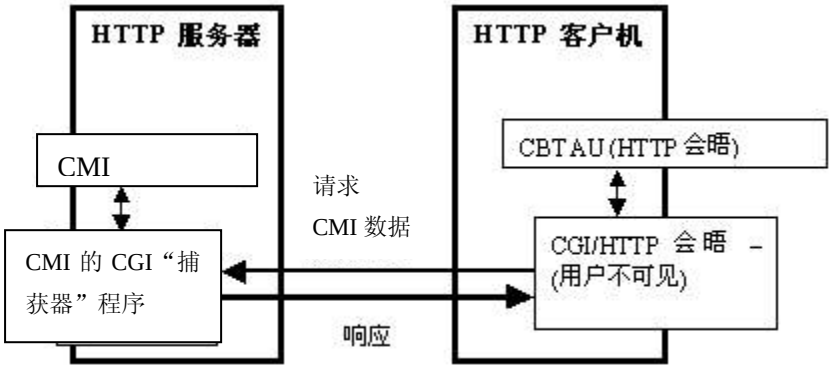

图 15 CMI/CBT 通信流

6.3.1 HTTP 通信

这部分描述了使用在 HICP (HTTP IEEE CMI Protocol) 中 HTTP 消息的格式。HICP 消息在 HTTP/1.0 消息的条款中有描述。HTTP 消息的详细描述不在本文档的范围之内。 对于 HTTP 消息的更详细描述可见 RFC1945,，Hypertext Transfer Protocol （超文本传输协议）--HTTP/1.0

HTTP 客户机 (万维网浏览器)和 HTTP 服务器 (万维网服务器)之间的通信是通过消息机制完成的。存在两种 HTTP 消息，

- 请求 ：由客户机发送
- 响应 ：由服务器发送

HICP 中的 HTTP 请求消息使用 POST 方法，“ content-type ” 为“ application/x-www-form-urlencoded ”。HICP 的 HTTP 响应消息的“ content-type ”是“text/plain”。

HICP 请求/发送消息数据包含在发送/请求消息（分别的）中的实体"entity-body"中。

名称/值对

消息数据格式遵循名为“名称/值对”的规定。名称/值对如下定义：

<名称>=<值>

名称代表了一个字段名，值表示了这个字段的内容。后面的部分描述了具体的格式。

6.3.2 HTTP 请求消息格式

Content-type: application/x-www-form-urlencoded
Request Method: POST

格式

表 8 给出了具体的格式：

表 8 HTTP 请求消息格式

<名称>	<值>
command=	<IEEE 命令 >&
version=	<IEEE 指定版本>&
session_id=	<唯一的学习段标识>&
AU_password=	<可分配单元密码(可选)>&
AICC_Data=	<(URL 编码的) IEEE 数据> <缓冲区结束>

其中：

- <*IEEE 命令* > = 任意有效 HTTP 命令(参见 HTTP IEEE CMI 协议命令)
- < *IEEE 指定版本* > = IEEE Spec Version (例如 1.9 版)
- <*唯一的学习段标识*> =*唯一的学习段标识符*(参见 5.1.2 节 IEEE 需要的参数)
- <*可分配单元密码*(*可选*)>= 可分配单元特定密码
- <*IEEE 数据*>=命令用的特定数据

使用规则：

- 上面所有的值都是 URL 编码的。
- 名称/值对可以任意顺序出现。
- 如果省略可选值，名称也应被省略。
- 每个参数的名称不区分大小写。

示例：

Command=GetParam&version=2.0&session_id=AX36&AICC_Data=

Command=PutParam&Version=2.0&Session_id=DAX36 &AICC_Data=[core]%0D%0A lesson_location+%3D+end%0D%0Alesson_status%3Dpass%0D%0Ascore%3D87%0D%0A time%3D00:23:15

注意：%0D 是回车的十六进制值，%0A 是换行的十六进制值，%3D 是等号的十六进制值，%26 是&的十六进制值。

6.3.3 HTTP 相应消息格式

Content-type: text/plain
Request Method: POST

格式

表 9 给出了具体的格式：

表 9 HTTP 响应消息格式

<名称>	<值>
error=	<IEEE 错误代码><CR>
error_text=	<IEEE 错误描述 (可选)><CR>
version=	<IEEE 指定版本(可选)><CR>
aicc_data=	<IEEE 数据> <缓冲区结束>

此处：

- <*CR*> = 回车和换行的字母(ASCII 13 10)
- < *IEEE 错误代码*> = IEEE HTTP 错误消息代码(见下节)
- < *IEEE 错误描述* > = IEEE HTTP 错误消息文本(见下节)
- <*IEEE 数据*>= PARAM.CMI 数据 (如果在请求中发出 GetParam 命令)

6.3.4 HTTP 通信错误消息

表 10 列出了通信错误消息：

表 10 HTTP 通信错误消息

错误代码	**错误文本**
0	Successful 成功
	Invalid Command 无效命令行
1	Invalid AU-password 无效可分配单元-密码
2	Invalid Session ID 无效学习段 ID 号
3	Command not supported 不支持的命令
4	<Undefined error> 没有定义的错误
5	

使用规则：

- Tab 和 space 允许出现在<名称>，“=”和<值>之前或之后。
- **aicc_data 中的 <值>** 数据以“=”后第一个的非空白字母开始，直到缓冲区结束。
- 所有其它的<**名称**>变量的<**值**>数据以“=”后第一个的非空白字母开始，直到回车换行之前最后一个的非空白字符结束。
- **<值>** 数据应该是纯文本（不是 URL 编码）
- **aicc_data** 只包括在对 GetParam 请求的响应中。
- 如果 返回 **aicc_data**，它必须是实体中的最后一个名称/值对。
- 名称/值对中名称的字母不区分大小写。
- 如果省略一个可选值，则它的名称也应被省略。

示例：

Error=0
error_text = successful
version= 2.0
aicc_data=[core]
Student_ID=B1781
Student_Name=Doe, John
CREDIT=C
Lesson_Location=
Lesson_Mode=Browse
Lesson_Status = Not Attempted
Score=
Time = 00:00:00
[Student_data]
max_time_allowed=00:45:00
time_limit_action=Exit,Message

6.3.5 HTTP CMI 协议命令

表 11 定义了 HTTP CMI　协议命令

表 11 协议命令

命令	功能	使用规则
GetParam	从 CMI 中获得输入数据。 (数据格式在 6.2 节中定义)	*必需。* 这个命令可能会被多次发送到 CMI 系统。 (注意：在可分配单元的学习段期间，从 CMI 系统收集的数据应该是相同的。)
PutParam	发送输出参数文件数据到 CMI 系统 (数据格式在 6.2 中定义)	*必需。* 这个命令可能多次使用。每一次使用，输出参数数据将被替换。CMI 只使用在 CBT 可分配单元学习段最后的 PutParam 内的数据。 （也就是说，这是一个 “覆盖”的操作） 使用多 **PutParam** 可以避免由于连接的结束或正常的学习段终止而引起的数据丢失。
PutInteractions	发送至少一行数据（也就是一个记录）	*可选。* 这个命令可以被多次调用。每次在可分配单元中调用这个命令，CMI 系统都会收集并存储新的数据内容。 (也就是说，这是一个“添加”操作)
ExitAU	结束一次可分配单元的学习段	*必需。* 这个命令仅在可分配单元学习结束时传送。

6.3.6 万维网启动参数

定义：

是基于 Web 的可分配单元需要使用的，课特有的启动参数。它们是字符序列，附加到文件名和 CMI-生成（请求）参数后面，用于在万维网环境中成功启动可分配单元。

这个数据在 URL 命令行的“？”分隔符后的询问部分。（参见 6.2.1.4 节）。“万维网启动参数”是独立的可分配单元所要求的附加参数。

数据格式：

数字和文字的。值可能区分大小写。字段标识符是 **web_launch。**

使用规则：

- 参数内容以“<参数名> = <参数内容>”的形式进行通信。
- 这些名称/值对由“&”隔开。
- 名称/值对可以以任意顺序。
- 参数名不区分大小写。
- 参数值可区分大小写。
- 所有的参数必须是 URL 编码的。

6.3.7 可分配单元_密码

定义：

发送到 CMI 系统的一个字符串，它使 CMI 系统可以认证一个可分配单元。

AU_password 是可选项，目的是为了确保安全。密码对于 AU 来说是特定的，而且将与 HICP 请求消息一同发送，因此 CMI 系统可以认证 AU 所做的请求。

CMI 系统核对 AU 文件的登陆内容。CMI 对这个登陆内容和 AU 发送过来的内容进行比较。

AU_ password 的值应该是通过系统管理员配置给各个 CBT 可分配单元的，而不应该是一个 AU 内嵌或“硬编码”的静态值。

数据格式：

数字和文字的。值是区分大小写的，并限制在 255 个字符内。字段标识符是 **AU_Password。**

示例：

rtjh4578gh

trust!1

6.4 CBT/CMI 通信信息

这些数据是标准课程为了执行功能而必须从 CMI 系统获取的，和在课程结束之后作为学生表现记录传递回 CMI 系统的。首先列出的是核心项，然后是根据字母顺序排列的其它项。每个组后面的是属于该组的关键字。

6.4.1 表格描述

- 强制(Obl)：
  - 必要项 (M)：CMI 系统必须提供这些数据。课程可以选择是否使用，但无论是用与否，它们都是存在的。
  - 可选项 (O)：CMI 系统可能提供这些数据。这些数据可能是课程为了更好地运行所需要的组或者关键字数据。课程必须为它们设置一个默认值以防 CMI 系统无法提供这些数据。
- 权限 (Prv)：
  - Get：课可以通过传递 GetParam 请求从 CMI 系统中得到值。
  - Put：课可以在 PutParam 中设置值。
  - Get/Put：课可能同时 get 和 put 一个值。

表 12 数据元素的组和关键字分配

组或关键词名称	数据元素名称	数据元素定义	强制	权限
[核心]	Core	可能被课使用的通用信息	M	-
学生编号	|--Student ID	唯一的字母-数字表示的代码/标识符，表示 CMI 系统中独立的一个使用者。	M	Get
学生姓名	|--Student Name	通常，使用学程登记表中学生的正式姓名。是完整的姓名，而不仅是名。	M	Get
课的位置	|--Lesson Location	这是传送给 CMI 系统关于学生最后一次使用这个可分配单元时的退出点	M	Get/ Put
学分	|--Credit	表明学生在本节课的学习（通过/失败和分数）是否得到 CMI 系统的学分	M	Get
课的状态	|--Lesson Status	这是由 CMI 系统确定的学生当前状况，当可分配单元启动时传送给它。	M	Get/ Put
课的状态，入口	|--Entry	说明了以前学生是否进入过这个可分配单元。	M	Get

组或关键词名称	数据元素名称	数据元素定义	强制	权限
课的状态，出口	|--Exit	表明学生如何以及为何离开本课.	M	Put
分数	|--Score	学生在本节课上次学习中的表现	M	-
分数	|--|--Raw	用数值形式表示的学生在课中的成绩。可能是没有处理的原始分数.	M	Get/ Put
分数	|--|--Max	学生获得的最高分数或总分。	O	Get/ Put
分数	|--|--Min	学生获得的最低分数	O	Get/ Put
时间	|--Total Time	学生学习这个可分配单元的累计用时	M	Get
时间	|--Session Time	学生学习这节课的时间	M	Put
模式	|--Lesson Mode	标识在启动后需要的课的行为。	O	Get
[核心_课]	Suspend Data	在以前的使用中，可分配单元生成的独特的信息，这些信息对于现在的使用也是必需的。	M	Get/ Put
[核心_提供者]	Launch Data	可分配单元创建时独特的信息，在每次使用中都是必要的。	M	Get
[注释]	Comments	学生在本次上课时记录下的注释.	O	Put
[评语]	Comments from LMS	教师对于学生的直接评语，当需要时，可分配单元可以提供给学生。	O	Get
[目标_状态]	Objectives	学生完成本节课每个教学目标的情况	O	-
J_标号.1	|--ID	开发者定义的，特定课的目标编号	O	Get/ Put
J_分数.1	|--Score	为掌握该目标学生每次学习得到的分数	O	-
J_分数.1	|--|--Raw	用数值形式表示的学生在课中的成绩。可能是没有处理的原始分数	O	Get/ Put
J_分数.1	|--|--Max	学生在此目标上测验的最高得分	O	Get/ Put
J_分数.1	|--|--Min	学生在此目标上测验的最低得分	O	Get/ Put
J_状态.1	|--Status	为掌握该目标学生每次学习后的状态	O	Get/ Put

组或关键词名称	数据元素名称	数据元素定义	强制	权限
[学生数据]	Student Data	基于学生表现定制一节课所需要的信息	O	-
掌握得分	|--Mastery Score	在课堂之外事先规定的及格分数。	O	Get
最大允许时间	|--Max Time Allowed	允许学生在当前课中学习的时间	O	Get
限时反应	|--Time Limit Action	当超过最大允许时间时，课要采取的行动	O	Get
[学生偏好]	Student Preference	学生给出的参数选择，会影响后续课	O	-
音频	|--Audio	声音开/关，音量控制	O	Get/ Put
语言	|--Language	信息发送所用的语言	O	Get/ Put
速度	|--Speed	内容传递步调	O	Get/ Put
文本	|--Text	文本可视化控制	O	Get
[交互]	Interactions	从学生到计算机的一组可辨别可记录的输入	O	-
交互编号	|--ID	由课程开发者决定的唯一的数字－字母标识	O	Put
目标编号	|--Objectives.n.ID	标明所有与本交互相关的目标	O	Put
交互时间	|--Time	标明交互对学生开放的时间	O	Put
交互类型	|--Type	标明记录的交互的种类	O	Put
正确反应	|--Correct Responses.n	期待学生在交互中作出的正确的反应	O	Put
反应种类	|--Pattern	定义学生可能的反应种类	O	Put
权重	|--Weighting	决定各个交互之间重要关系的因素	O	Put
学生反应	|--Student Response	描述计算机可识别的学生反应	O	Put
学生反应	|--Result	对学生反应作出的评价	O	Put
时间间隔	|--Latency	从开始交互到得到反应之间的时间间隔	O	Put

6.4.2 [Core]（核心）

名称	数据元素	数据元素定义	强制	权限
[核心]	Core	可能被课使用的通用信息	M	-

定义

核心 是对 **GetParam** 调用应答所需要的组，它应该包含以下关键字：

Student_ID
Student_Name
Lesson_Location
Credit
Lesson_Status
Score
Time

它可以包含以下的关键字：

Mode

核心是对 **PutParam** 调用应答所需要的组，它应该包含以下关键字：

Lesson_Location
Lesson_Status, flag
Score
Time

GetParam 调用举例

[core]
student_ID = CAM-1942
student_name = McArthur, Christopher A. Jr.
lesson_location=0
credit=credit
lesson_status=complete
time=00:23:15
score=93

6.4.2.1 Student ID（学生编号）

名称	数据元素	数据元素定义	强制	权限
学生编号	|--Student ID	唯一的字母-数字表示的代码/标识符，表示 CMI 系统中独立的一个使用者。	M	Get

定义

参见标准草案 CELTS-20.1 ， 第 5.2.1 节

格式

CMIIdentifier

示例

student_id=Jack_Hyde1
student_ID = JQH-1942
STUDENT_ID= JACK1991-3

6.4.2.2 Student Name（学生姓名）

名称	数据元素	数据元素定义	强制	权限
学生姓名	|--Student Name	通常，使用学程登记表中学生的正式姓名。是完整的姓名，而不仅是名。	M	Get

定义

参见标准草案 CELTS-20.1 ， 第 5.2.2 节

格式

CMIString255

示例

Student_name=Whiplash, William R.

STUDENT_NAME= Grey, Jane S.

student_name = McArthur, Christopher A. Jr.

6.4.2.3 Lesson_location（课程位置）

名称	数据元素	数据元素定义	强制	权限
课的位置	|--Lesson Location	这是传送给 CMI 系统关于学生最后一次使用这个可分配单元时的退出点	M	Get/ Put

定义

参见标准草案 CELTS-20.1，第 5.2.4 节

数据格式

CMIString255

与具体实现有关。CMI 系统只负责保存这些数据并在学生再次进入课程时将它们返回课。课程传递给 CMI 系统的任何信息都被传回。此格式与任何课程要求的和建立的格式相匹配。

6.4.2.4 Credit（学分）

名称	数据元素	数据元素定义	强制	权限
学分	|--Credit	表明学生在本节课的学习（通过/失败和分数）是否得到 CMI 系统的学分	M	Get

定义

参见标准草案 CELTS-20.1 ， 第 5.2.5 节

格式

CMIVocabulary

只是首字符有意义，不区分大小写。

示例

Credit=c

CREDIT = NO-CREDIT

credit=N

6.4.2.5 Lesson Status（课的状态）

名称	数据元素	数据元素定义	强制	权限
课的状态	|--Lesson Status	这是由 CMI 系统确定的学生当前状况，当可分配单元开始时传送给它。	M	Get/ Put
课的状态，入口	|--Entry	说明了以前学生是否进入过这个可分配单元。	M	Get
课的状态，出口	|--Exit	标识学生为何以及如何离开本节课	M	Put

定义

参见标准草案 CELTS-20.1 ，第 5.2.6 节

入口标志

参见标准草案 CELTS-20.1，第 5.2.7 节

出口标志

参见标准草案 CELTS-20.1 ，第 6.1.3 节

格式

一个字、字符，或短语代表了在 CMIVocabulary 表中定义的数据状态，也可以在后面跟上逗号或是其它的表示状态的字或字符。每个字只有第一个字母有意义。注意如果状态标志缺省表示学生正常重新进入。

示例 1: entry

Lesson_Status = failed

; 学生上次没有通过这门课程

示例 2: entry

Lesson_Status = N,A

; 学生第一次进入这门课程，需要"A" 标志和"Not Attempted"状态。

示例 3: entry

课程_状态 = p

; 学生前一次已经通过了这门课程，没有 A 标志意味着学生不是第一次进入这门课程。

; 没有 R 标志意味着学生通过这门课程时是正常离开的（也就是没有挂起标志）

示例 4: entry

lesson_status=i,r

; 学生没有完成这门课程。当他离开时，会产生一个挂起标志，因此需要一个恢复标志

示例 5: entry

lesson_status = p,a

; 学生已经表明掌握了本课的内容，例如通过了预先测试。Ab 标志意味着这是他第一次进

；入这节课。

示例 6: entry

lesson_status=i

；学生没有完成这门课程。当它离开的时候，产生了注销或者其它标志，没有产生挂起标志。

示例 7: exit

Lesson_Status = incomplete,logout

；学生在没有完成课程的情况下退出了

示例 8: exit

Lesson_Status = incomplete, suspend

；学生在没有完成课程的情况下离开，学生可能还打算返回本课。

示例 9: exit

Lesson_Status = not attempted

；学生浏览了课程的某些部分，不打算正式学习而离开

示例 10: exit

lesson_status = p,l

；学生通过了课并希望退出课程。

6.4.2.6 Score（分数）

名称	数据元素	数据元素定义	强制	权限
分数	|--Score	学生在本节课上次学习中的表现	M	-
分数	|--|--Raw	用数值形式表示的学生在课中的成绩。可能是没有处理的原始分数。	M	Get/ Put
分数	|--|--Max	学生获得的最高分数或总分。	O	Get/ Put
分数	|--|--Min	学生获得的最低分数	O	Get/ Put

定义

参见标准草案 CELTS-20.1，第 5.2.9 节

格式

CMIDecimal

如果包括了最高分和最低分，那么就要用逗号把它们隔开。顺序应该是：**原始分数，最高分数，最低分数**。

使用规则

学生第一次尝试时，**分数＝""**。即后面的分数是空的。对于多次尝试而言，分数记载的是最后一次的成绩。

示例

score= 79

score=1, 2

score = 3.5, 8.1, -1

6.4.2.7 Time（时间）

名称	数据元素	数据元素定义	强制	权限
时间	|--Total Time	学生在学习这个可分配单元的累计用时	M	Get
时间	|--Session Time	学生学习这节课的时间	M	Put

定义

Time 与数据元素 **Total Time** 和 **Session Time** 有关，参见标准草案 CELTS-20.1，第 5.2.10 节

CMI 系统负责统计总时间，而不是跟踪每一个学习段的时间。可分配单元负责统计学习段时间并且在结束时传回 CMI 系统。

格式

CMITimespan

6.4.2.8 Mode（模式）

名称	数据元素	数据元素定义	强制	权限
模式	|--Lesson Mode	说明关于学生用来改变可分配单元行为的相关信息。	O	Get

定义

参见标准草案 CELTS-20.1 ， 第 5.2.11 节

格式

CMIVocabulary

示例

Mode = B

Mode = review

6.4.3 [Core_Lesson] 核心_课

名称	数据元素	数据元素定义	强制	权限
【核心_课】	Suspend Data	在以前的使用中，可分配单元生成的独特的信息，这些信息对于现在的使用也是必需的。	M	Get/ Put

定义

参见 **Suspend Data**，标准草案 CELTS-20.1 ， 第 5.3 节

格式

CMIString4096

核心_课的格式可以是课唯一的。关于数据唯一的限制是：

- 数据必须采用 ASCII 格式传递，课可能随后将它们转化为内部格式。

- **Core Lesson** 数据必须被限制在 4096 字节以内。

6.4.4 [Core_Vendor]核心_提供者

名称	数据元素	数据元素定义	强制	权限
[核心_提供者]	Launch Data	课创建时产生的独特的信息，在每次使用中都是必要的。	M	Get

定义

参见标准草案 CELTS-20.1 ， 第 5.4 节

格式

CMIString4096

6.4.5 [Comments]（注释）

名称	数据元素	数据元素定义	强制	权限
[注释]	Comments	学生在本次上课时记录下的注释	O	Put

定义

参见标准草案 CELTS-20.1 ， 第 6.3 节

位置示例

<l.frame12>

<L.page 36>

<L.fuel3-21>

<L.Fuel part 3: interaction 24>

位置参数表明了学生在何时以及课的哪个位置书写注释。位置与其它需要的信息一起被放置在注释限制符之内。

位置可以和课元素或部分相对应。课元素是拥有唯一名称（ID）的可分配单元的任意划分的部分。

示例

[Comments]

<1><L.apu.intro>为什么标志和我的航班的标志不一致<E.1>

<2><L.apu.q3>我不明白为什么 B 是对的，难道不应该首先检查防火栓么？ <E.2>

[Comments]

<1> Electrical 这个单词在这里被拼错了<E.1>

；没有位置标志的话，这样的评价是没有意义的

6.4.6 [Comments]（评语）

名称	数据元素	数据元素定义	强制	权限
[评语]	Comments from LMS	教师对于学生的直接评价，当需要时，可分配单元可以提供给学生。	O	Get

定义

参见标准草案 CELTS-20.1，第 5.5 节

格式

CMIString4096

示例

[Comments]

<1>这节课之后的训练课程将在星期四 13:15 而不是起初通知的 9:00。<e.1><2>如果你已经通过了练习部分，请你跳过这节课的练习部分<E.2>

[Comments]

<1> <L.Frame23>关于这个主题更多的信息可以在课本的 3.12 节找到<e.1>

6.4.7 [Objectives_Status]（目标_状态）

名称	数据元素	数据元素定义	强制	权限
【目标_状态】	Objectives	学生完成本节课每个教学目标的情况	O	-

定义

参见标准草案 CELTS-20.1 ， 第 5.7 节

本组包括以下的关键字

J_ID.1

J_Score.1

J_Status.1

示例

[objectives_status]

J_ID.1 = APU1684

j_status.1 = passed

6.4.7.1 J_ID.1（J_标号.1）

名称	数据元素	数据元素定义	强制	权限

J_标号.1	|--ID	开发者定义的，特定课的目标编号	O	Get/ Put

定义

J_ID 应该是一个由开发者定义的、针对单个课的目标标志

关键字格式

每一个 **J_ID** 关键字都有一个用来区分的扩展标号，它应该是一个点号和随后的一个 0 到 9999 之间的数字。数字首位的 0 必须去掉（例如，0009 是不符合条件的，应该直接使用 9）。

数据格式

CMIIdentifier

使用规则

Objectives_Status 组可以包含多个**标号**，但是每个标号都必须要有唯一的扩展标号。

因为每个目标**标号**的值是在 CBT 课件中内部定义的字符串，CMI 系统应该有储存和引用那些课定义**标号**的手段。

示例

J_ID.1 = A1373

6.4.7.2 J_Score.1（J_分数.1）

名称	数据元素	数据元素定义	强制	权限
J_分数.1	|--Score	为掌握该目标学生每次学习得到的分数	O	-
J_分数.1	|--|--Raw	用数值形式表示的学生在课中的成绩。可能是没有处理的原始分数	O	Get/ Put
J_分数.1	|--|--Max	学生在此目标上测验的最高得分	O	Get/ Put
J_分数.1	|--|--Min	学生在此目标上测验的最低得分	O	Get/ Put

定义

J_Score 应该是学生上次尝试掌握目标的表现。目标可能包括与它们相关的分数，如果有的话，这个关键字能够捕捉到那些数据。

如果有最高分和最低分，那么它们之间应该用逗号分开。这个分数反映了学生上次尝试目标的情况。

关键字格式

分数关键字应该有一个扩展标号以标识它的唯一，它应该是一个点号和随后的一个 0 到 9999 之间的数字。数字首位的 0 必须去掉（例如，0009 是不符合条件的，应该直接使用 9）。

数据格式

一个或多个 CMIDecimal 类型的数字，按照原始分、最高分、最低分的顺序用逗号分开。

使用规则

扩展部分允许将分数和一个 **J_ID** 相关联。例如，**.1** 的分数就是针对 **J_ID.1** 标识符的一组成绩。

示例

J_Score.1 = 2

；在学生的上次尝试中，学生得到了 2 分

J_Score.1 = 2, 3, 0

J_Score.2 = 4,,0

J_ID.1= First

J_Score.1 = 87

J_ID.2= Second

J_Score.2 = 3,5

；在上一次的尝试中，学生得分 3，也可能是 5

J_ID.1= 1

J_Score.1 = 87, 100, 0

；对目标 1 有原始、最高、最低分值。

J_ID.2= 2

J_Score.2 = 3

6.4.7.3 J_Status.1（J_分数.1）

Name	Data element	Data element definition	Obl	Prv
J_Status.1	|--Status	为掌握该目标学生每次学习后的状态。	O	Get/ Put

定义

参见 Status，标准草案 CELTS-20.1

关键字格式

每一个状态关键字都有一个用来区分的扩展标号，它应该是一个点号和随后的一个 0 到 9999 之间的数字。数字首位的 0 必须去掉（例如，0009 是不符合条件的，应该直接使用 9）。

数据格式

CMIVocabulary

使用规范

状态必须有和 **J_ID** 相同的扩展名称，这标志着对应目标的状态。

对于一个目标永远不能放置两个或以上的状态关键字。但是，如果一个目标有多于一个状态，那么课程将自动假定第一个状态是正确的。

如果一个状态没有对应的 **J_ID**，那么状态将被忽略。

如果目标 ID 没有与状态相关联，那么对应状态将是 **Not Attempted（没有尝试）**。

示例

j_id.3 = 1987

j_status.3=p

j_id.6 = 1942

J_STATUS.6 = incomplete

J_ID.92 = 1847
J_Status.92 = P

6.4.8 [Student_Data] （学生数据）

名称	数据元素	数据元素定义	强制	权限
[学生数据]	Student Data	基于学生表现定制一节课所需要的信息	O	-

定义

参见标准草案 CELTS-20.1，第 5.8 节

此组包括以下的关键字：

Mastery_Score

Max_Time_Allowed

Time_Limit_Action

6.4.8.1 Mastery_Score（掌握得分）

名称	数据元素	数据元素定义	强制	权限
掌握得分	|--Mastery Score	在课堂之外事先规定的及格分数。	O	Get

定义

参见标准草案 CELTS-20.1，第 5.8.2 节

格式

CMIDecimal

示例

mastery_score = .75

Mastery_Score = 100

6.4.8.2 [Max_Time_Allowed]（最大允许时间）

名称	数据元素	数据元素定义	强制	权限
最大允许时间	|--Max Time Allowed	允许学生在当前课中学习的时间	O	Get

定义

参见标准草案 CELTS-20.1 ，第 5.8.3 节

数据格式

CMITimespan

示例

max_time_allowed = 0:14:30.5
Max_Time_Allowed = 2:03:00
MAX_TIME_ALLOWED = 1:09:00

6.4.8.3 Time_Limit_Action（限时反应）

名称	数据元素	数据元素定义	强制	权限
限时反应	|--Time Limit Action	当超过最大允许时间时，课要采取的行动	O	Get

定义

参见标准草案 CELTS-20.1，第 5.8.4 节

格式

CMIVocabulary

对于 HTTP 通信，只有逗号前后每个单词的首字母才被考虑，不区分大小写。

示例

time_limit_action = Exit, Message
；课向学生给出超时的消息，然后退出或停止。

Time_Limit_Action=E,N
；课直接退出或停止，不给学生信息。学生直接跳到 CMI 环境中。

time_limit_action = c,n
；当学生超过给定时间时，系统不做提示并继续该课。

TIME_LIMIT_ACTION = continue, message
；学生在超过给定时间时，得到系统提示，但是可以继续该课。

6.4.9 [Student_Preference]（学生偏好）

名称	数据元素	数据元素定义	强制	权限
[学生偏好]	Student Preference	学生给出的参数选择，会影响后续课	O	-

定义

参见标准草案 CELTS-20.1 ，第 5.10 节

本组有以下的一些关键字：

Audio
Language
Speed
Text

6.4.9.1 Audio（音频）

名称	数据元素	数据元素定义	强制	权限
音频	|--Audio	声音开/关，音量控制	O	Get/ Put

定义

参见标准草案 CELTS-20.1 ，第 5.10.1 节

格式

CMISInteger

示例

audio= -1

AUDIO = 33

6.4.9.2 Language（语言）

名称	数据元素	数据元素定义	强制	权限
语言	|--Language	信息发送所用的语言	O	Get/ Put

定义

参见标准草案 CELTS-20.1 ，第 7.10.2 节

格式

CMIString255

数字-字符串，可以包括空格。

6.4.9.3 Speed（速度）

名称	数据元素	数据元素定义	强制	权限
速度	|--Speed	内容传递步调	O	Get/ Put

定义

参见标准草案 CELTS-20.1，第 5.10.4 节

格式

CMISInteger

示例

speed= -100

；如果一个系统只有快、中、慢三种速度的话，那么使用任何正数（+1 或以上）都将会代表快速。

SPEED = 33

6.4.9.4 Text（文本）

名称	数据元素	数据元素定义	强制	权限
文本	|--Text	文本可视化控制	O	Get

定义

参见标准草案 CELTS-20.1，第 5.10.5 节

格式

CMISInteger

示例

Text = -1

text=0

TEXT = 1

6.4.10 [Interactions]（交互）

名称	数据元素	数据元素定义	强制	权限
【交互】	Interactions	从学生到计算机的一组可辨别可记录的输入	O	-

定义

参见标准草案 CELTS-20.1，第 8.6 节

Interactions 必须采用 **PutInteractions** 命令传递。 **交互**不应该在组中或者是以关键字模式被传递，也不能像本章描述的其他数据元素那样进行 **PutParam** 操作。**交互**必须被放入由逗号分割的表格式中，采用 URL 编码。表中的每一行都是一个记录，传送的字段将采用以下的标志符作为首行的名称：

Interaction_ID

Objective_ID

Time

Type_Interaction

Correct_Response

Weighting

Student_Response

Result

Latency

典型的 **PutInteractions** 命令将传递以下的信息（为了清楚起见，下面的例子没有使用 URL 编码）：

"time","interaction_id","objective_id","type_interaction",

"correct_response","student_response","result","weighting",

"latency"

"15:14:23",37,ft1016, C,A,C,W,,00:00:3

"15:14:23",38,ft2223,t,t,t,,,00:00:01

"15:14:23",39,ft1134,C,B,B,C,,00:00:02

"15:14:23",40,ft1156,C,C,C,C,,00:00:04

在表格式中，列可以以任何顺序。第一行，或者是标题行，决定下面各行值的顺序。所有的列都是可选的，所有的值也是可选的。即使该列存在，其中的内容也可能为空。

PutInteractions 命令将传递完整的记录或行。每条命令可以传送一行或者整个表格。

PutInterations 命令都是一个添加操作。

6.4.10.1 Interaction_ID（交互编号）

名称	数据元素	数据元素定义	强制	权限
交互编号	|--ID	由课程开发者创建的唯一的数字－字母标识	O	Put

定义

参见标准草案 CELTS-20.1，第 8.6.1 节

格式

CMIIdentifier

6.4.10.2 Objectives_ID（目标编号）

名称	数据元素	数据元素定义	强制	权限
目标编号	|--Objectives ID	标明所有与本交互相关的目标	O	Put

定义

参见标准草案 CELTS-20.1，第 8.6. 2 节

格式

CMIIdentifier（标准标识符）

如果有多个目标与同一个交互关联，必须用逗号把这些目标分隔开。在这种逗号分割的格式中，逗号间的字段都要用引号括起来。

6.4.10.3 Time（交互时间）

名称	数据元素	数据元素定义	强制	权限
交互时间	|--Time	标明交互对学生开放的时间	O	Put

定义

参见标准草案 CELTS-20.1 ，第 8.6.3 节

格式

CMITime

6.4.10.4 Type_Interaction（交互类型）

名称	数据元素	数据元素定义	强制	权限
交互类型	|--Type	标明记录的交互的种类	O	Put

定义

参见标准草案 CELTS-20.1，第 8.6.4 节

格式

只用 CMIVocabulary 成员中的首字母是重要的，不区分大小写。

示例

"true-false"

"multiple choice"

-- 这将被理解为类型匹配，因为只有此类中的首字母是有效的

"C"

"F"

"Performance"

6.4.10.5 Correct_Responses（正确反应）

名称	数据元素	数据元素定义	强制	权限
正确反应	|--Correct Responses	期待学生在交互中作出的正确的反应	O	Put

定义

参见标准草案 CELTS-20.1，第 8.6.5 节

格式

CMIFeedback

示例

交互种类	表示
Choice	"b; d" --学生选择 b 或者 d 都将被视为正确
Choice	"{b,d}" -- 学生只有同时选择 b 和 d 才能被视为正确
Matching	"1.c; 2.b; 3.a; 4.d" -- 如果学生作出了以上的任何一种匹配，都将被视为正确
Matching	"{1.c,2.b,3.a,4.d}" -- 只有当学生作出所有的匹配才能被视为正确
Matching	"{3.4,1.6,5.2}" --只有当学生作出所有的匹配才能被视为正确
Matching	"{a.e, d.g, c.f, b.h};{a.h, d.g, c.f, b.h}" --学生必须至少作出四个匹配才能被视为正确，但是他把 a 与 e 或 h 配对也将被视为正确

混合示例	"<case> Washington" "2300-2400" -- 这些可以是填空题的正确答案。如果它代表的是一个技能题，那它们之间必须有两个连字符号 "2300 -- 2400" "b,c,e,a,d" --一个顺序问题，这是一个简单答案 "23291" --注意对于有些问题而言，大的整数之中不能有逗号，例如 23,291。将被视为两个不同的响应

6.4.10.6 Weighting（权重）

名称	数据元素	数据元素定义	强制	权限
权重	|--Weighting	决定各个交互之间重要关系的因素	O	Put

定义

参见标准草案 CELTS-20.1 ， 第 8.6.6 节

格式

CMIDecimal

6.4.10.7 Student_Response（学生反应）

名称	数据元素	数据元素定义	强制	权限
学生反应	|--Student Response	描述计算机可识别的学生反应	O	Put

定义

参见标准草案 CELTS-20.1，第 8.6.7 节

格式

CMIFeedback

示例

Matching 示例	"1.2" "{1.c,2.b,3.a,4.d}" "{3.4,1.6,5.2}" "{a.e, d.g, c.f, b.h}" "{12.2, 11.3, 10.11}"
Mixed 示例	"1.2" "1.2,1.3"

	"Washington" "C" "b,c,e" "23291" -- 注意对于有些问题而言，大的整数之中不能有逗号，例如 23,291。将被视为两个不同的反馈

6.4.10.8 Result（结论）

名称	数据元素	数据元素定义	强制	权限
结论	|--Result	对学生反应的评价	O	Put

定义

参见标准草案 CELTS-20.1，第 8.6.7.1 节

格式

CMIVocabulary

示例

"c"（正确）

"c,c,w,c,c"（正确、正确、错误、正确、正确）

"unanticipated response, correct, correct, wrong, c"
（未知答案、正确、正确、错误、正确）

"neutral"（中性）

"Correct"（正确）

6.4.10.9 Latency（时间间隔）

名称	数据元素	数据元素定义	强制	权限
时间间隔	|--Latency	从开始交互到得到反应之间的时间间隔	O	Put

定义

参见标准草案 CELTS-20.1，第 8.6.8 节

格式

CMITimespan

示例

"00:00:23"

-- 学生需要 23 秒反应

"00:23:00"

--学生需要 23 分钟反应

"00:00:03.4"

-- 学生需要 3.4 秒反应

第 7 章 交换用课程集合数据模型

课程

课程是一个完整的教学单元，可以被一个或多个学生使用，提供为某个学科或完成一组相关任务所需要的知识或技能。它由课和测验（可分配单元）及相关学习目标构成，是某个课程体系的一部分。

在本标准中，课程是CMI所能管理的最大单元。

目的

定义CMI结构互换格式的目的，是为了简化将课程从一个CMI系统移动到另一个CMI系统的过程。这些基本的数据结构还可以作为课程开发工具的输出。通常用任务分析工具或教学系统设计工具创建课程结构。通过使用标准化方法描述一个课程，可以使课程的实际实现变得更为简单容易。

在课程移动之后，对其重新审阅和修改的工作还是需要的。标准互换文件节省了将草稿输入产生新课程所需要的大量的人工时间。

组织

移动课程所必需的信息逻辑上可以分为三部分：

描述

这是关于课程和它的组成成分的信息。它包括的数据诸如课程及其组成成分的标题和文字描述等。

结构

该数据描述了课程的组织形式。它定义课程如何分解，不同的课程部分怎样组织起来。它还包括说明学习目标彼此之间的关系，以及学习目标和课程各部分间关系的信息。

次序

这个数据描述了学生所看到的课程内容的次序。描述先后顺序的机制能够支持线性顺序、完全由学生控制的内容次序、以及根据算法决定的更为复杂的内容次序。

7.1 基本概念

对课程结构的描述需要回答这样一个问题，“为了按照设计的方式向学生呈现训练材料，CMI 系统需要哪些信息？”

7.1.1 列表：隐含次序

本标准文档假设包含全部课程和课程组的列表隐含地定义了次序。该列表是一系列可用表格表述的数据元素。

通过在课程中说明每节课（或可分配单元）的先修条件，可以显性定义次序。先修条件必须在学生选修或上课之前设定，每节课在课程结构中都有一个位置。

示例：假定一个课程有 6 节课。课的次序可以用简单列表表述。

```
Lesson 1
Lesson 2
Lesson 3
Lesson 4
Lesson 5
Lesson 6
```

7.1.2 先修条件：显性次序

为了显性定义次序，假设第六课的先修条件是要求学生必须先完成第 5 课，而第 5 课要求先通过第 4 课，第 4 课又要求完成第 3 课等等。其结果就导致课程以从第 1 课到第 6 课的顺序呈线性表述。

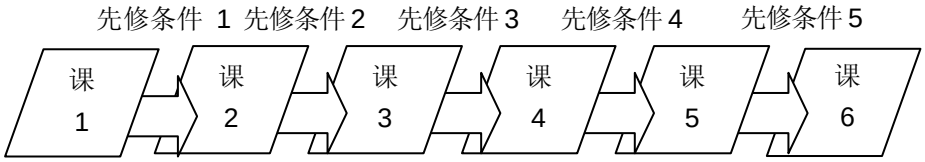

图 15 课的先后次序

这些课之间的关系还可以用表 13 描述：

表 13 先修条件表

可分配单元	先修条件
第 1 课	无
第 2 课	第 1 课
第 3 课	第 2 课
第 4 课	第 3 课
第 5 课	第 4 课
第 6 课	第 5 课

当然，即使存在先修条件，有时候仍然需要学生自己来选择次序。比如有三节课要求同样的先修条件，那么学生在满足先修条件后，就要从三节课中任选一节课学习。
先修条件可以按已完成的课程，或已掌握的目标定义。

状态

在前面介绍 CMI 与课之间通信的章节中，曾经提到在学生离开课时要将有关信息传回 CMI 系统，这些信息包括课状态（*Lesson Status*）和目标状态（*Objectives.Status*）。
这些状态可以用来确定对于课程中的每个结构成分（可分配单元和块区），先修条件是否满足。

7.2 课程要素

课程要素有三类：

- 可分配单元（课）
- 块
- 教学目标（简单和复杂）

块和可分配单元是结构元素，教学目标不是。

```
Course elements//课程元素
    Structure elements//结构元素
        Blocks//块
        Assignable units = lessons//可分配单元=课
             Blocks//块
                  Blocks//块
                       Assignable units//可分配单元
             Blocks//块
                  Assignable units//可分配单元
                  Blocks//块
                       Assignable units//可分配单元
    Objectives//教学目标
```

7.3 复杂度级别

“规范”定义了表述课程结构的三种复杂度级别。从第 1 级到第 2 级，再到第 3a 级或 3b 级，复杂程度的增加会导致：

在输入数据后，花在 CMI 系统修改上的功夫将减少。

更完整地描述开发者关于课程材料如何使用的意图。

第一级

这是最简单一级。它描述课程的内容------课或可分配单元，根据可分配单元和块，定义课程的层次结构。学生学习课程的次序隐含在结构表示中。该描述不能将任何次序强加于学生。

第二级

这个级别允许为每个结构元素------可分配单元或块[1]说明一先修条件。每一个先修条件的状态很简单：完成或未完成。学生学习课程的顺序由先修条件强制决定。在这一级还能够判别一个结构元素的状态如何影响另一个结构元素，为此可以设立独立的可分配单元作为预测验。这样，某个可分配单元（如预测单元）的完成可以导致另外一个可分配单元（如教学单元）变成“通过”状态。

第 3A 级

[1]在这一级，块的完成状态有省缺规则决定，一个块特定的完成要求在 3a 以上级的结构定义。

这个级别允许为每个课程要素定义更加复杂的先修条件和完成要求。结构元素的状态可以用逻辑表达式描述。

第 3B 级

在这一级可以为结构元素加入教学目标，这些目标既可以作为先修条件，也可以作为可分配单元和块的完成状态，每个块或可分配单元的多个先修条件可以用含有与（&）和或（|）的逻辑表达式定义。

约束条件

支持第二级的 CMI 系统必须能够支持第一级的所有功能，支持第 3A 级或第 3B 级的 CMI 系统必须能够支持第二级的所有功能，支持第 3 级的 CMI 系统必须能够支持第 3A 级和第 3B 级的所有功能。只能支持第 2 级和 3A 或 3B 级部分功能的 CMI 系统只能算具有附加功能的第一级系统。

7.4 课程描述数据

下面以列表的形式给出“规范”所定义的课程描述数据，具体的数据元素的解释信息请参见“规范”文本。

表 14 课程描述数据

名称	中文名称	含义	表值	级别	数据类型
Properties	属性	课程整体信息	S	1	-
|--Creator	创作者	销售商名或课程作者名	*	1	CMIString255
|--Identifier	课程编号	课程唯一的标签	S	1	CMIIdentifier
|--System	写作系统	用于开发课程的主要写作系统	S	1	CMIString255
|--Title	课程名	课程的一般名称	S	1	CMIString255
|--Level	级别	反映课程结构和编列复杂度	S	1	CMIString255
|--Max Block Members	最大块成员数	最复杂块中成员数目	S	1	CMIInteger
|--Max Objective Members	最多目标成员数	最复杂的教学目标关系中成员个数	S	1	CMIInteger
|--Total AU's	可分配单元总数	课程中可分配单元的总数	S	1	CMIInteger
|--Total Blocks	块总数	课程中块的总数目	S	1	CMIInteger
|--Total Objectives	教学目标总数	课程中教学目标（简单和复杂）的总数	S	1	CMIInteger
|--Total Complex Objectives	复杂目标总数	课程中复杂目标总数	S	1	CMIInteger
|--Version	版本	课程编列数据所基于的 IEEE CMI 标准的版本号	S	1	CMIString255
名称	**中文名称**	**含义**	**表值**	**级别**	**数据类型**

|--Description	描述	关于课程的文本信息	S	1	CMIString4096
|--Max Normal	最多未完成单元数	需要记分，同时进行学习的可分配单元最大数目	S	4	CMIInteger
Structure	结构	课程组织和编列的信息	S	1	-
|--Block	块	课程中的最大结构元素。总是包含其他课程元素	+	1	-
|--|--Identifier	块标识符	开发者为每块创建的唯一标识	S	1	CMIIdentifier
|--|--System Identifier	块系统标识符	由 CMI 系统创建的在课程内的唯一标识符	S	1	CMISIdentifier
|--|--Title	块名称	块的名称	S	1	CMIString255
|--|--Description	块说明	块内容的文字概述	S	1	CMIString4096
|--|--Prerequisites	块先修条件	在开始某块学习前学生必须达到的要求	*	2	CMILogic
|--|--Completions	块完成要求	学生为了通过块的学习必须做什么	+	2	--
|--|--|--Requirement	要求	可以判断真或假的表达式	S	2	CMILogic
|--|--|--Status	状态	当要求表达式为真时给学生学分。	S	2	CMIVocabulary
|--|--|--Next AU	下一个可分配单元	在状态为真的情况下强制性安排	S	2	CMISIdentifier
|--|--|--Return to	返回可分配单元	在离开下一个可分配单元之后强制性安排	S	2	CMISIdentifier
|--Assignable Unit	可分配单元	与可分配单元相关的信息	*	1	-
|--|--Identifier	可分配单元编号	开发者为可分配单元创建的唯一标识	S	1	CMIIdentifier
|--|--System Identifier	可分配单元系统标识符	由 CMI 系统创建的在课程中的唯一标识	S	1	CMISIdentifier
|--|--Title	可分配单元名称	可分配单元、块、目标或复杂目标的名称	S	1	CMIString255
|--|--Description	可分配单元说明	.可分配单元内容的文本概述	S	1	CMIString4096
|--|--Type	可分配单元类型	开发者确定的可分配单元的类别	S	1	CMIString255
|--|--Launch Line	启动命令	成功地启动可执行程序所需要的字符串。	S	1	CMIString255
名称	**中文名称**	**含义**	**表值**	**级别**	**数据类型**

|--|--File Name	文件名	包含某节课内容的文件名全称	+	1	CMIString255
|--|--Max Score	最高成绩	该节课报告的最高成绩	S	1	CMIDecimal
|--|--|--Mastery Score	及格分数	学生通过该节课所需的最低成绩	S	1	CMIDecimal
|--|--|--Max Time Allowed	最大允许时间	学生学习这节课最多允许的时间	S	1	CMITimespan
|--|--Time Limit Action	限时反应	当最大允许时间超出时，课所要做的动作	S	1	CMIVocabulary
|--|--System Vendor	系统供应商	创建该节课的写作系统	S	1	CMIString255
|--|--Launch Data	启动数据	课的设计所要求的信息	S	1	CMIString4096
|--|--Prerequisites	先修条件	用逻辑表达式表示在开始可分配单元学习前，学生必须完成什么。	S	2	CMILogic
|--|--Completions	完成要求	学生为了通过可分配单元的学习必须做什么	+	2	--
|--|--|--Requirement	要求	可以判断真或假的表达式	S	2	CMILogic
|--|--|--|--Status	状态	当要求表达式为真时给学生学分。	S	2	CMIVocabulary
|--|--|--Next AU	下一个可分配单元	在状态为真的情况下强制性安排	S	2	CMISIdentifier
|--|--|--Return to	返回可分配单元	在离开下一个可分配单元之后强制性安排	S	2	CMISIdentifier
|--|--|--Embedded Objective	嵌入教学目标	在可分配单元中的教学目标	*	3B	CMISIdentifier
Objective	目标	可测量的学习目标	*	3B	-
|--Identifier	目标编号	开发者给每个目标分配的唯一标识	S	3B	CMIIdentifier
|--System Identifier	目标系统标识符	由 CMI 系统创建的在课程内的唯一标识符	S	3B	CMISIdentifier
|--Title	目标名	目标的通用名称	S	3B	CMIString255
|-- Description	目标说明	目标的文字概述	S	3B	CMIString4096
|--Member IDs	目标成员标识	由 CMI 分配给目标中每个课程元素的唯一标识	*	3B	CMISIdentifier
|--Completions	目标完成要求	学生为了达到目标要求必须做什么	+	3B	--
|--|--Requirement	要求	可以判断真或假的表达式	S	3B	CMILogic
|--|--Status	状态	当表达式为真时给学生学分。	S	3B	CMIVocabulary

7.5 结构示例

这一节给出一系列的例子解释“规范”所描述的课程结构组织原则。

约定

在这些结构的例子中，用点分割开的类和数据元素与用层次标识的元素是等同的。例如：
Course.Structure.Block
等同于 Course
Structure
Block

在例子中有意忽略了许多数据元素，这是为了便于理解，强调例子中的重点部分。

任何在圆括弧（）中的数据或信息只是起注释的作用，并不是数据元素值的一部分。

7.5.1 简单课程示例

这是一个包含七节课的简单课程。它们没有被组织成组（块）。

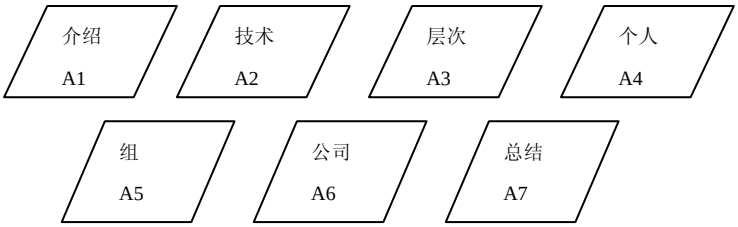

图 16 简单课程

例子数据说明

下面的纪录描述了上面的图表。每一个可分配单元标识符都是一个“系统标识符”——由系统为课程的输出生成文件时分配的标识符。为了强调结构，其他一些数据元素被省略了。

注意在下面的例子中一些数据元素被有意省略，这样有助于突出重点。

```
Course.Structure
    AssignableUnit
        SystemIdentifier: A1
        Title: Introduction
    AssignableUnit
```

```
        SystemIdentifier: A2
        Title: Technology
    AssignableUnit
        SystemIdentifier: A3
        Title: Layers
    AssignableUnit
        SystemIdentifier: A4
        Title: Individual
    AssignableUnit
        SystemIdentifier: A5
        Title: Group
    AssignableUnit
        SystemIdentifier: A6
        Title: Corporation
    AssignableUnit
        SystemIdentifier: A7
        Title: Summary
```

也可以以下面方式描述：

```
Course.Structure.Block
    SystemIdentifier: B0
    Title: The root (The course)
        AssignableUnit.SystemIdentifier: A1 (Introduction)
        AssignableUnit.SystemIdentifier: A2 (Technology)
        AssignableUnit.SystemIdentifier: A3 (Layers)
        AssignableUnit.SystemIdentifier: A4 (Individual)
        AssignableUnit.SystemIdentifier: A5 (Group)
        AssignableUnit.SystemIdentifier: A6 (Corporation)
        AssignableUnit.SystemIdentifier: A7 (Summary)
```

7.5.2 含有块的课程举例

同样是这七节课，但是其中五节课分成了两组（块）。

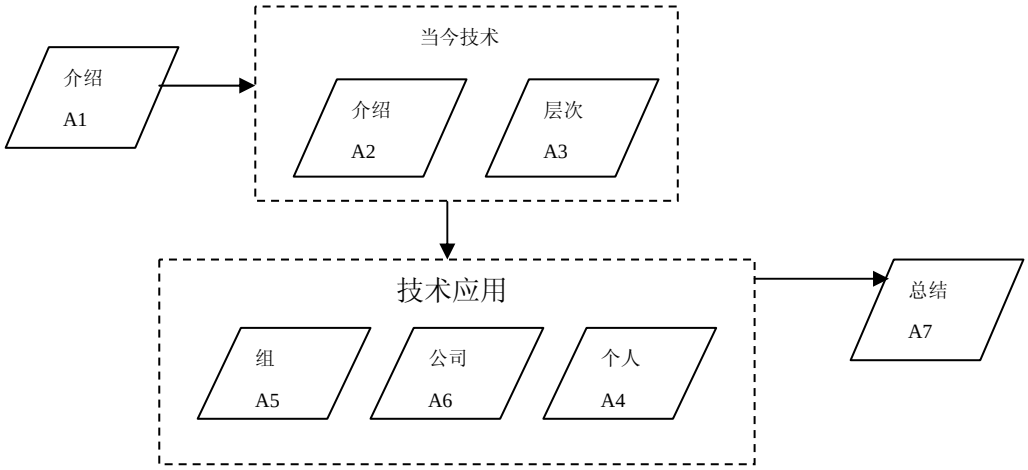

图 17 含有块的课程

例子数据说明

下面的记录反映和解释了上述图表。 每一个块标识符和成员标识符都是系统标识符——由系统为课程的输出生成文件时分配的标识符。为了强调结构，其他一些数据元素被省略了。

```
Course.Structure
    AssignableUnit
        SystemIdentifier: A1
        Title: Introduction
    Block
        SystemIdentifier: B1
        Title: Technology Today
            AssignableUnit
                SystemIdentifier: A2
                Title: Technology
            AssignableUnit
                SystemIdentifier: A3
                Title: Layers
    Block
        SystemIdentifier: B2
        Title: Technology Applications
            AssignableUnit
                SystemIdentifier: A4
                Title: Individual
```

```
            AssignableUnit
                SystemIdentifier: A5
                Title: Group
            AssignableUnit
                SystemIdentifier: A6
                Title: Corporation
```

或者：

```
Course.Structure
   Block
      SystemIdentifier: B0
      Title: The root – The course
         AssignableUnit.SystemIdentifier: A1 (Introduction)
         Block.SystemIdentifier: B1 (Technology Today)
         Block.SystemIdentifier: B2 (Technology Applications)
         AssignableUnit.SystemIdentifier: A7 (Sum)
   Block
      SystemIdentifier: B1
      Title: Technology Today
         AssignableUnit.SystemIdentifier: A2 (Technology)
         AssignableUnit.SystemIdentifier: A3 (Layers)
   Block
      SystemIdentifier: B2
      Title: Technology Applications
         AssignableUnit.SystemIdentifier: A4 (Individual)
         AssignableUnit.SystemIdentifier: A5 (Group)
         AssignableUnit.SystemIdentifier: A6 (Corporation)
```

7.5.3 两个块的课程

这是一个简单的课程，可以用三种方式描述，第一种描述是图表，第二种描述是表格，第三种描述是实例。

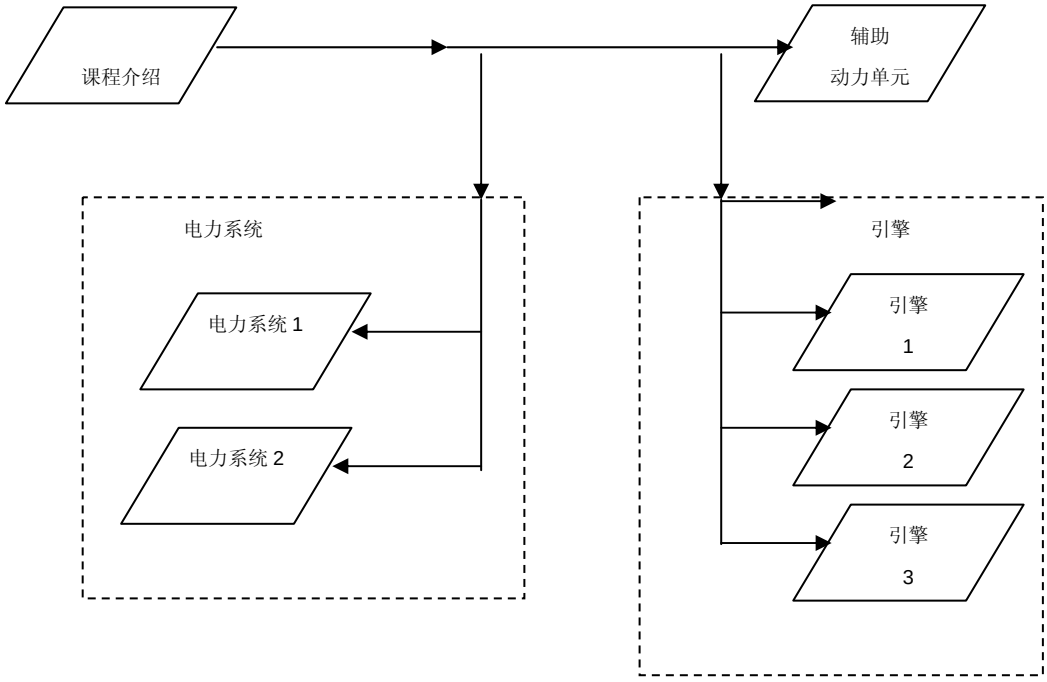

图 18 简单课程的图表表示

表格说明

下面的表格反映了上面的图表。表格中的每一个课程元素都采用它的“标题”标识。

表 15 介绍课程的表格

块	成员	成员	成员	成员
根	介绍	辅助动力单元	电力系统	Engine 引擎
电力系统	电力系统 1	电力系统 2		
引擎	引擎 1	引擎 2	引擎 3	

包含两个块的课程描述

下面的记录反映了上面的图表和表格。 每一个块标识符和成员标识符都是系统标识符——由系统为课程的输出生成文件时分配的标识符。为了强调结构，其他一些数据元素被省略了。

```
Course.Structure
   AssignableUnit
      SystemIdentifier: A1
      Title: Introduction
   AssignableUnit
      SystemIdentifier: A2
      Title: Auxiliary Power Unit
   Block
      Identifier: B1
      Title: Electrical Power
         AssignableUnit
            SystemIdentifier: A3
            Title: Electrical, Part 1
         AssignableUnit
            SystemIdentifier: A4
            Title: Electrical, Part 2
   Block
         Identifier: B2
         Title: Engine
            AssignableUnit
               SystemIdentifier: A5
               Title: Engine, Part 1
            AssignableUnit
               SystemIdentifier: A6
               Title: Engine, Part 2
            AssignableUnit
               SystemIdentifier: A7
               Title: Engine, Part 3
```

或：

```
Course.Structure
   Block
      Identifier: B0
      Title: The course -- Root
         AssignableUnit.SystemIdentifier: A1 (Introduction)
         AssignableUnit.SystemIdentifier: A2 (Aux Power Unit)
```

```
        Block.SystemIdentifier: B2 (Electrical Power)
        Block.SystemIdentifier: B3 (Powerplant – Block 2)
Block
        Identifier: B1
        Title: Electrical Power
        AssignableUnit.SystemIdentifier: A3 (Electrical, Part
        1)
        AssignableUnit.SystemIdentifier: A4 (Electrical, Part
        2)
Block
        Identifier: B2
        Title: Engine
        AssignableUnit.SystemIdentifier: A5 (Engine, Part 1)
        AssignableUnit.SystemIdentifier: A6 (Engine, Part 2)
        AssignableUnit.SystemIdentifier: A7 (Engine, Part 3)
```

7.5.4 编列

当输出或输入一门课程时，按照该标准，可以在各个可分配单元、块、目标的状态的基础上做出两种类型的决定。课程编列数据描述课程元素的使用情况，以便：

确定学生到何时已满足进行一项学习活动的先修条件。（他准备好了吗？）

确定学生到何时已经完成了一项学习活动。（学习完成了吗？）

图 19 排序编列决定

对学生在一门课程中的学习活动顺序的描述可以使用先修条件或完成需求，据此，可以定义相当复杂的课程顺序。

以下几节分析了“先修条件”和“完成需求”是怎样确定一门课程的顺序的。

7.5.5 逻辑表达式

逻辑表达式可以描述先修条件和完成需求。

逻辑表达式由课程元素（块、可分配单元、目标）的状态（完成、未完成，等等）及逻辑运算符（&，|，~）组成，其值为“真”或“假”。一个特殊的逻辑表达式是单个单词“never”。这是为了防止学生在纪录有效的情况下，以其他方式（常规、复习、浏览）进入课程。

逻辑运算符

逻辑操作符描述了课程元素是如何结合的，从而决定先修条件或完成需求的结果是“真”还是“假”。下表列举了常用的逻辑运算符。

运算符含义	符号
与	&
或	|
Never	never
非	~
相等	=
不相等	<>
组或集合	{ }
集合元素分隔符	,
complete X number out of a set	X*{ }
先计算	( )

表 16 逻辑运算符

可分配单元（课）状态

逻辑表达式使用课程元素的状态，比如“**Completed**（完成）”是一个状态。课的状态由课程内部的逻辑设计决定。CMI 系统可以使用由课返回的状态确定其他课或块的先修条件以及其他块或复杂目标的完成状态。

每节课有六种可能的状态。

- Passed（通过）
- Completed（已完成）
- Browsed（浏览）
- Failed（失败）
- Not attempted（未尝试）
- Incomplete（未完成）

在逻辑表达式中，结构元素可以用等号和相应的状态连结。例如，表达式“A15= Passed”表明：如果要表达式为“真”，A15 的状态必须为“Passed”。然而，如果没有显式地标识，如果 A15 在表达式中没有与等号为伍，这五种状态就简化为两种状态：完成（真）或未完成（假）。如下表示：

完成（真）

Passed（通过）

Completed（完成）

未完成（假）

Failed（失败）

- Browsed（浏览）
- Not attempted（未尝试）
- Incomplete（未完成）

目标状态

教学目标状态也可以用于逻辑表达式。简单的目标状态由课决定。复杂目标的状态则由 CMI 确定。与可分配单元一样，五种可能的目标状态通常也简化为“完成”和“未完成”两种。

说明

逻辑表达式中的课程元素，如果状态没有显式地声明，就是“完成”或“未完成”两种状态之一，这两个状态对应于传统的逻辑数值“真”和“假”。下面列出的操作符可以用来和课程元素一起创建逻辑表达式。

与	当由符号“&”间隔的所有元素都完成时，表达式才是“complete（完成）”状态。 **A34 & A36 & A38** 可分配单元 34，36 和 38 都是完成状态时（Passed 或 Completed），这一组才被认为是“完成”状态。
或	如果被操作符“|”间隔的课程元素中有一个的状态为“通过”，则整个表达式的值为“真”。 **A34=P | A36=P | A38=P** 如果 34，36 和 38 中有一个状态为“通过”，则上述表达式的值为“complete”。
Never	特殊表达。如果先修条件文件中的第二个元素状态为“never”，则第三个元素所指明的模式下不能使用在第一项指出的课程元素。 **Element Identifier: A34** **Requirement: never** **Mode: Review** 可分配单元 A34 在“回顾”（Review）模式中不能进入使用。

非	如果后续的元素或表达式值为“完成”（真），则整个表达式的返回值为“未完成”（假）。反之亦然。如果后续的元素或表达式值为“未完成”（假），整个表达式的值为“完成”（真）。 **Element Identifier: A34** **Requirement: ~A35** 学生可以进入 A34 单元，只要 A35 单元状态为“未完成”（也就是说，A35 状态必须为“未完成”，“失败”，或者“未尝试”几种之一）。如果可分配单元 A35 的状态是“完成”，学生就不能进入 A34 单元。
相等	如果操作符两侧的元素值相同，则该表达式返回值为“真” **Element Identifier: A34** **Requirement: A33=Passed** 如果学生已经通过了 A33 单元，他就可以进入 A34 单元学习。
不相等	仅当操作符两侧的元素值不同时，表达式的返回值为“真”。 **Element Identifier: A34** **Requirement: A35<>Passed** 只要没有通过 A35 单元的学习，学生就可以进入 A34 单元。 注意此操作符和“非”（not）操作符之间的区别。~A35 又可表示为（A35<>Passed & A35<>Completed）。
集合	一组被逗号隔开，以花括号括起来的课程元素。集合和块有所不同，它只是为先修条件文件所定义，对课程的结构不起作用。 **{A34, A36, A37, A39}** 可分配单元 A34,A36,A37 和 A39 都是这一集合的一部分。
分隔符	逗号用来分隔集合中的各个成员。集合中的每一个成员的状态可以表示为布尔型——“完成”或“未完成”。 **{A34, A36, A37, A39}** 可分配单元 A34,A36,A37 和 A39 被逗号隔开。

X*	X 是一个整数。这个操作符的意思是当集合中的 X 或 X 以上个成员完成时，表达式才为真。 **Element Identifier: A38** **Requirement: 3*{A34, A36, A37, A39}** 可分配单元“34，36，37，39”中任何三个或多于三个成员的状态必须为“完成”方可进入单元 38。
evaluate 1st	处于圆括弧中的逻辑表达式优先计算，圆括号可以嵌套[2]。 **Element Identifier: A39** **Requirement: A34 & A35 | A36** 在上述表达中，仅仅完成 A36，学生即可进入 A39 的学习。 **Element Identifier: A39** **Requirement: A34 & (A35 | A36)** 在此表达中，因为添加了圆括号，这样就至少要完成两个单元的学习（只完成 A36 是不够的）方可进入 A39。

示例

这些数据元素来自先修条件或完成需求类：

第二级

```
A23
```

要使表达式为真，学生必须完成或通过可分配单元 23。

第二级

```
A23=passed
```

要使表达式为真，学生必须要通过可分配单元 23，仅仅完成可分配单元还不够。

3A 级

```
A23 & A28
```

要使表达式为真，学生必须要完成或通过可分配单元 23 和 28。

3A 级

```
3*{A23, A25, A26, A28, A29}
```

要使表达式为真，学生必须至少完成括号中 5 节课中的三节课。

3A 级

```
3*{A23, (A25 & A26) A28, A29}
```

在这个例子中，可分配单元 25 和 26 一起成为集合的一个成员，因此学生必须完成 4 个可分配单元才能使表达式为真，比如，如果学生只是完成了 A23, A25,和 A28, 表达式不会为真。

[2] 操作符和括号的运算优先级同 C 程序设计语言。

3B 级

```
~J31
```

如果教学目标 31 没有完成也没有通过，需求为真。也就是，如果教学目标 31 的状态为 Incomplete, Failed, Browsed, 或 Not Attempted，表达式为真。

3B 级

```
~(J31=F)
```

如果学生在教学目标 31 没有失败，表达式为真。也就是说，如果教学目标 31 的状态为 Fail，表达式的值为假。

3A & 3B 级

```
A14 & ~J15
```

要使表达式为真，学生必须完成可分配单元 14，并且还没有完成教学目标 15. 如果学生已经掌握了教学目标 15，表达式就为假。如果学生还没有完成 14 课，表达式也为假。

7.5.6 先修条件

先修条件可用来设置和描述非常复杂的课编列系统。先修条件也是课程交换文件中编列描述的基础。

有的时候，需要阻止学生进入某节课因为他没有达到某些先修条件。可以为课程中的每个块或每个可分配单元分别放置这类约束，这些约束可以使单个先修条件（2 级），也可以是一组先修条件（3A 级）。

数据元素

所有的先修条件的表达中，第一个数据元素为课程的结构元素，第二个为先修条件，第三个为进入该课程元素的进入模式。

系统产生的系统标识符用于表达“结构元素”。“先修条件”是一个逻辑表达式，根据课程元素的状态决定学生是否可以开始块或可分配单元的学习。

下面的例子用到了 2 级先修条件。

```
Course
   Structure
      Block
         AssignableUnit
            Prerequisites: CMISIdentifier
```

下面的例子用到了 3A 级先修条件

```
Course.Structure.Block
   AssignableUnit
         Prerequisites: CMILogic
```

示例 1: 3A 级

在可分配单元 A4 的先修条件种，出现了课程结构元素 A1、A2 和 A3。

```
Course
   Structure
      Block.SystemIdentifier: B1
         AssignableUnit.SystemIdentifier: A4
               Prerequisites: A1 & A2 & A3
```

这意味着可分配单元 A1、A2、A3 都必须是“**Completed**”状态时，学生才能够开始学习可分配单元 A4，换句话说，学生必须在第一、二、三节课得到“通过”或“完成”状态，才能学习第四节课。

示例 2: 3A 级

```
Course.Structure
   Block
      System Identifier: B2
      Assignable Unit
         System Identifier: A4
         Prerequisites: A1=P & A2=P & A3=P
```

这表示在学生开始学习第四节课之前，其第一、二和第三节课的学习必须达到了“Passed”状态。

示例 3: 3A 级

在一个强调教学目标的课程中，可能会出现下面的先修条件：

```
Course.Structure
   Block
      System Identifier: B0
      Assignable Unit
         System Identifier: A4
         Prerequisites: J1 & J2 & J3
```

这意味着学生必须完成或通过这三个教学目标才能进入第 4 可分配单元的学习。

示例 4: 3A 级

要解释下列表达式，必须是 3A 级系统

```
Course.Structure
   Block.SystemIdentifer: B12
      Assignable Unit
         System Identifier: A31
         Prerequisites: A30 & (A23 | A28)
```

则表示可分配单元 A31 定义了先修条件，学生必须完成 A30，并且还要完成 A23 或 A28。

示例 5: 第三级

要解释下列表达式，必须是第三级系统

```
Course
   Structure
      Block
         System Identifier: B14
         Assignable Unit
              System Identifier: A24
              Prerequisites: ~(J13 & J14 & J15)
```

这表示如果教学目标 13、14、15 没有完成的话，第 24 节课就会启动，并且只完成这些目标中的一两个并不能阻止学生进入 24 课，只有这三个目标都完成的时候才能防止正常进入 24 课。

示例 6 第二级

这门课只有三节课，A1、A2 和 A3，学生必须依次学习，下面的图示和先修条件表定义了课程的编列顺序。

图 19 简单课程

表 17 简单课程先修条件

先修条件表	
课程元素	**先修条件**
A1	无
A2	A1
A3	A2

A1 没有先修条件，A2 和 A3 有先修条件，所以先学 A1，掌握了 A1 之后，只有一节课的先修条件得以满足，就是 A2，所以 A2 第二个学，在掌握了 A2 后，必须学 A3，因此这张表会强制学生按顺序学习这几节课。

```
Course
   Structure
         AssignableUnit
            SystemIdentifier: A1
         AssignableUnit
            SystemIdentifier: A2
            Prerequisite: A1
         AssignableUnit
            SystemIdentifier: A3
            Prerequisite: A2
```

示例 7　3A 级

这也是有三节课的另一门课，这门课有不同的先修条件。

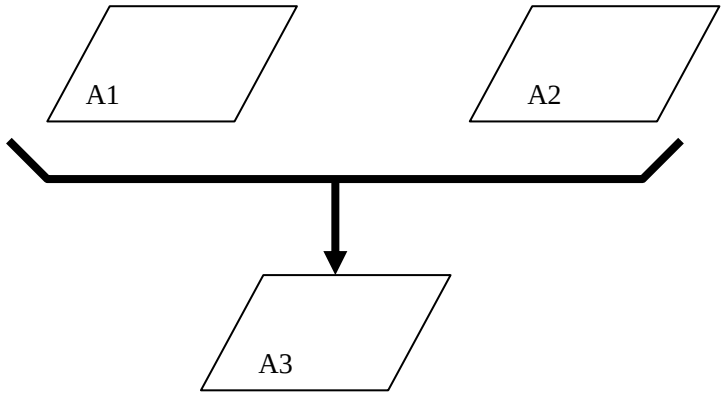

图 20 三节课的课程

在这个例子中，A1 或 A2 没有先修条件，所以学生可以随便先学这两节课之一，但是，他只能在掌握了 A1 和 A2 之后，才能学习 A3，如下面的先修条件所示：

表 18 三节课和先修条件

先修条件表	
课程元素	**先修条件**
A1	无
A2	无
A3	A1 & A2

```
Course
     Structure
            AssignableUnit
                  SystemIdentifier: A1
            AssignableUnit
                  SystemIdentifier: A2
```

```
AssignableUnit
      SystemIdentifier: A3
      Prerequisite: A1 & A2
```

示例 8 第二级

这个例子说明如何用简单的先修条件定义相对复杂的学生浏览模式。这门课有 10 节课。有三节课必须先学，这三节课的学习顺序随便，在学完这三节课后，学生可以在另外四节课中任选一节课学习，在学完了这四节课之后，学生才可以学剩下的三节课，并且学习顺序也随意。

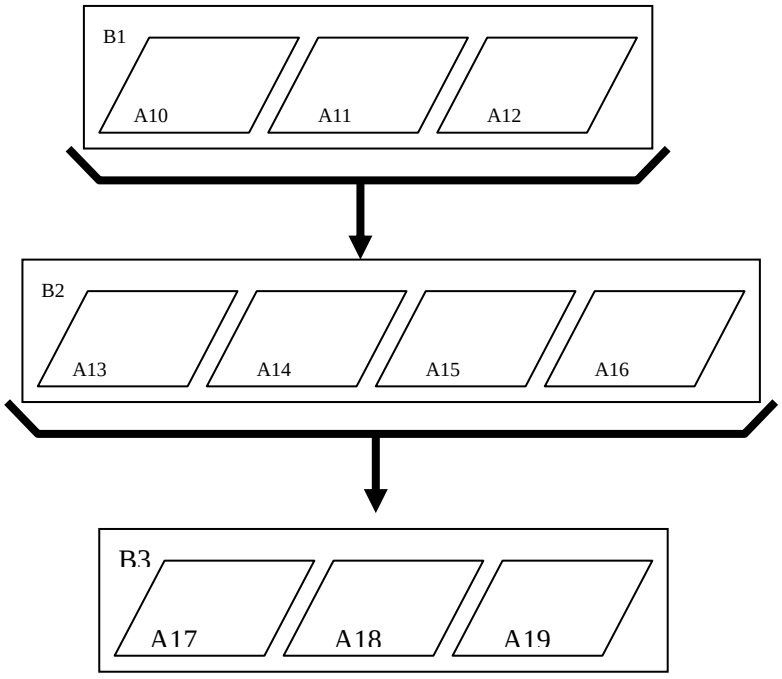

图 21 较大课程

这个例子包括块，前三节课在块 1(B1)，后面的四节课在块 2(B2)，最后三节课在块 3(B3)。下面的先修条件用块代替了单节课。

表 19 较大课程的先修条件

先修条件表	
课程元素	**先修条件**
B1	无
B2	B1
B3	B2

一般来说，一个块被认为通过了，就是当其所有成员都通过了之后，在这个例子中，每个块中的课可以以任何次序学习，但是块必须按顺序学习。例如，学生可以随时学习 A10, A11, 或 A12，但是他必须完成了 B1 块中的每节课(A10, A11, 和 A12)之后才能学习 A14。

```
Course.Structure
        Block
              SystemIdentifier: B1
              AssignableUnit.SystemIdentifier: A10
              AssignableUnit.SystemIdentifier: A11
              AssignableUnit.SystemIdentifier: A12
        Block
              SystemIdentifier: B2
              Prerequisite: B1
              AssignableUnit.SystemIdentifier: A13
              AssignableUnit.SystemIdentifier: A14
              AssignableUnit.SystemIdentifier: A15
              AssignableUnit.SystemIdentifier: A16
        Block
              SystemIdentifier: B3
              Prerequisite: B2
              AssignableUnit.SystemIdentifier: A17
              AssignableUnit.SystemIdentifier: A18
              AssignableUnit.SystemIdentifier: A19
```

示例 9 第二级

这个例子是一个有先修条件的简单课程结构，有三个块：电子、能源工厂和燃料，9 节课。学生可以随时选择任一块，但是在一个块内，必须按顺序学这些课。

例如，当选择了能源工厂时，学生必须先完成能源工厂的“燃料”课，然后完成“油”，完成“气体动力”，最后完成“过程”，学生可以在能源工厂学完之前选学其他块，但是当他返回到能源工厂块时，还必须继续按顺序学习。

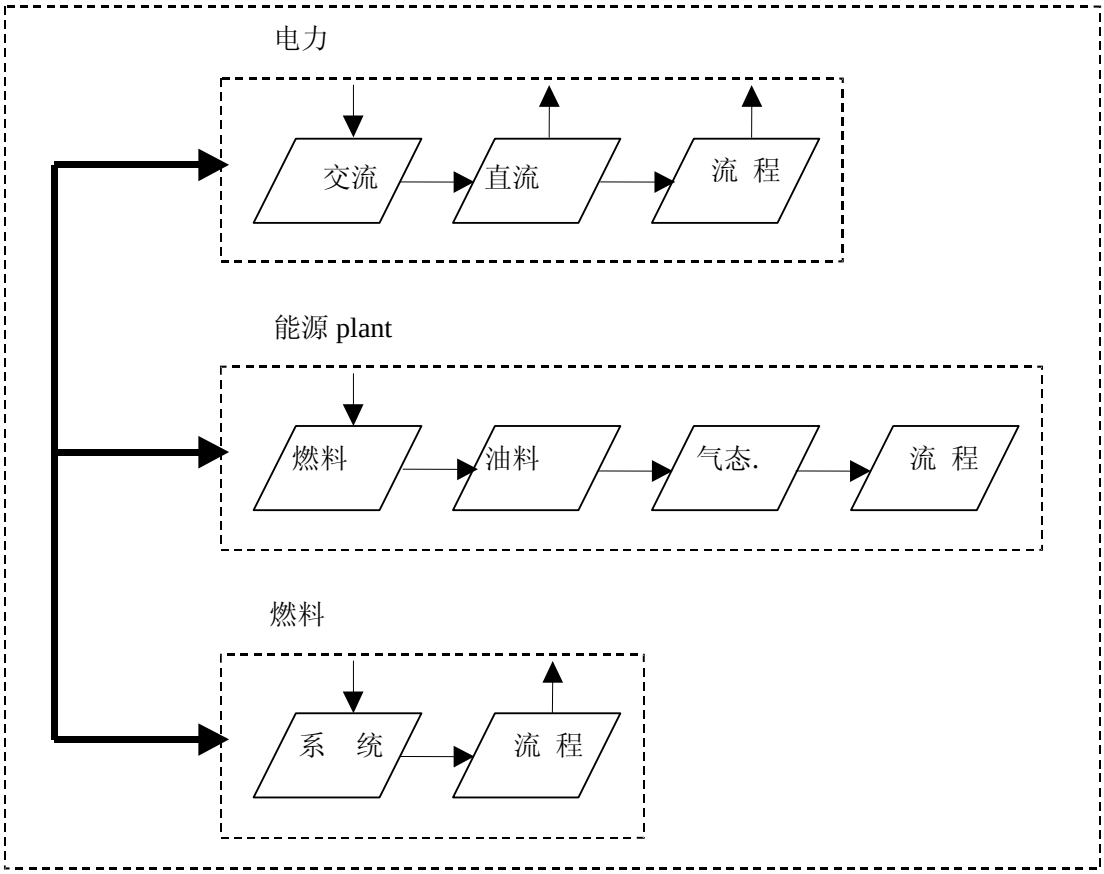

图 22　9 节课的课程

表 20 先修条件表

结构元素 **(块或可分配单元)**	**先修条件** **(块或可分配单元)**
电（块)	无
交流电	无
直流电	交流电
电流过程	直流电
能源工厂（块）	无
能源工厂燃料	无
能源工厂油	能源工厂燃料
能源工厂气体动力	能源工厂油
能源工厂过程	能源工厂气体动力
燃料（块)	无
燃料系统	无
燃料过程	燃料系统

按照这种先修条件要求的实例化数据表示为：

```
Course
     Structure
                Block
                     SystemIdentifier: B1
                     Title: Airplane Electrical System
                     AssignableUnit
                          SystemIdentifier: A1
                          Title: Electrical AC
                     AssignableUnit
                          SystemIdentifier: A2
                          Title: Electrical DC
                          Prerequisite: A1
                     AssignableUnit
                          SystemIdentifier: A3
                          Title: Electrical Procedures
                          Prerequisite: A2
                Block
                     SystemIdentifier: B2
                     Title: Power Plant
                     AssignableUnit
                          SystemIdentifier: A4
                          Title: Power Plant Fuel
                     AssignableUnit
                          SystemIdentifier: A5
                          Title: Power Plant Oil System
                          Prerequisite: A4
                     AssignableUnit
                          SystemIdentifier: A6
                          Title: Power Plant Pneumatics
                          Prerequisite: A5
                     AssignableUnit
                          SystemIdentifier: A7
                          Title: Power Plant Procedures
                          Prerequisite: A6
                Block
                     SystemIdentifier: B3
                     Title: Airplane Fuel System
                     AssignableUnit
                          SystemIdentifier: A8
                          Title: Fuel System
                     AssignableUnit
                          SystemIdentifier: A9
                          Title: Fuel System Procedures
                          Prerequisite: A8
```

7.5.7 完成需求

虽然每节课和每个目标的状态经常由课本身的逻辑设计所决定，但是也并不总是这样。例如，有一个对学生进行预先测验的可分配单元，如果学生在预先测验中表明已经掌握了其中的一些目标，学生就可能会因此获得该节课的部分学分，甚至全部学分，虽然他还没有看这节课一眼。换句话说，CMI 系统有时也可以根据课程元素之外的一些因素来确定该元素的状态。类似地，块和复杂目标的状态也是根据其他元素定义的。所以，块和复杂目标的状态应该由 CMI 系统来决定。

完成需求数据是设计用来明确说明一个可分配单元，块或目标何时能够得到某个状态，何时该状态不同于缺省状态。这就是例外文件。

缺省状态	**块** 块的状态是由它的所有成员的状态共同决定的。除非定义了完成需求，否则，当块的所有成员的状态是“Completed”时，这个块的状态就是“Completed”。 当一个或多个块成员的状态是“Not Attempted”，或者并不是所有成员的状态都是“Completed” 或“Passed”时，该块的状态就是“Incomplete”。 如果一个块的所有成员的状态都是“Not Attempted”，则这个块的状态就是“Not Attempted”。
	复杂目标 当复杂目标的成有成员的状态是“Completed”时，该复杂目标的状态就是“Completed”。 当一个或多个复杂目标成员的状态是“Not Attempted”，或者并不是所有成员的状态都是“Completed” 或“Passed”时，该复杂目标的状态就是“Incomplete”。 如果一个复杂目标的所有成员的状态都是“Not Attempted”，则这个复杂目标的状态就是“Not Attempted”。
	课 课的状态是在学生离开课时决定的。另外，CMI 系统也可以通过比较课的得分和课的及格分数，来决定课的状态是“通过”还是“失败”
	简单目标 简单目标的状态，在可分配单元将它的状态信息发送给 CMI 系统时确定的

例外文件中的每一组数据元素（这里称作记录）分别定义了 CMI 系统如何确定一个可

分配单元、块或者目标的状态。

决定每个课程元素的状态需要几个逻辑表达式。例如，仅仅对一节课定义 Passed, Failed, Completed, Incomplete 四种状态就需要 4 个完成需求记录。

这些记录的顺序是重要的。为了决定课的状态，CMI 系统按照这些逻辑表达式在例外文件中出现的顺序依次计算其值，第一个计算为“真”的表达式决定了该节课的状态。

下面给出的是完成需求，第二级，一个可分配单元完成需求的数据结构。

```
Course
     Structure
          Block
               Assignable Unit
                    System Identifier: CMISIdentifier
                    Completions
                        Requirement: CMILogic (CMISIdentifier)
                        Status: CMIVocabulary
                    Completions
                        Requirement: CMILogic (CMISIdentifier)
                        Status: CMIVocabulary
```

下面给出的是完成需求，第三级，一个可分配单元完成需求的数据结构。

```
Course.Structure.Block.AssignableUnit
     System Identifier: CMISIdentifier
     Completions
            Requirement: CMILogic
            Status: CMIVocabulary
     Completions
            Requirements: CMILogic
            Status: CMIVocabulary
```

实例纪录

下面的纪录取自假想数据集。

示例 1　3A 级

```
Course.Structure
     Block
          SystemIdentifier: B4
```

```
        Completions
              Requirement: A9 & A10 & A11 & A12
              Status: Pass
        AssignableUnit.SystemIdentifier: A9
        AssignableUnit.SystemIdentifier: A10
        AssignableUnit.SystemIdentifier: A11
        AssignableUnit.SystemIdentifier: A12
```

当所有成员状态是Passed 或Completed 时，块的状态为Passed. 这个例子将缺省状态假设显性的定义了。

示例 2　3A 级

```
Course.Structure.Block
    AssignableUnit
        SystemIdentifier: A3
        EmbeddedObjective: J31
        EmbeddedObjective: J32
        EmbeddedObjective: J33
        Completions
              Requirement: J31 & J32 & J33
              Status: Passed
```

显性完全需求最常用在基于教学目标的课件中，为了按照所掌握的目标决定每个课程元素的完成状态，就需要 Completions 数据类。

在上面这个例子中，掌握了目标 J31, J32 和 J33，学生就可以在可分配单元 A3 获得 Passed 状态。

示例 3：　第 2 级

```
Course.Structure.Block.AssignableUnit
    SystemIdentifier: A4
    Completions
          Requirement: A1
          Status: Passed
```

如果可分配单元A1的状态为完成或通过，可分配单元A4的状态就是通过，如果学生随后进入了A4单元, CMI必须向此单元发送“Passed”课状态（lesson status）。

示例 4：　第 2 级

```
Course.Structure.AssignableUnit
    SystemIdentifier: A4
    Completions
```

```
          Requirement: A1
          Status: Passed
    Completions
          Requirement: A2
          Status: Passed
    Completions
          Requirement: A3
          Status: Passed
```

如果可分配单元A1 或 A2 或 A3 的状态是完成或通过，那么可分配单元A4 的状态就是通过。如果纪录是顺序评判的，那么先为真的将决定A4的状态，比如A1 是 Passed，那么A2 和 A3 是通过还是失败，是否被学习过，都不再重要。

这个完成需求如果用 3A 级表示纪录会更少些。

示例 5： 3A 级

```
Course.Structure.AssignableUnit
     SystemIdentifier: A4
     Completions
          Requirement: A1 | A2 | A3
          Status: Passed
```

这个例子表达的内容和上面第 2 级的例子一样。

示例 6： 第 2 级

```
Course.Structure
     Block
          SystemIdentifier: B4
          Completions
               Requirement: A3=Passed
               Status: Passed
          AssignableUnit
               SystemIdentifier: A3
          AssignableUnit
               SystemIdentifier: A4
          AssignableUnit
               SystemIdentifier: A5
```

如果可分配单元 A3 的状态是“Passed”，那么整个第 4 块（不管它的其它成员）的状态就是通过。

示例 7： 第 3 级

```
Course.Structure
     Block
          SystemIdentifier: B13
          Completions
            Requirement: J23=P & J24=P & J25=P & J26=P
            Status: Completed
```

块 13 的状态不依赖于它的成员（可分配单元和其他块）是否完成，只在于目标是否达到。

示例 8： 3A 级

```
Course.Structure
     Block
          SystemIdentifier: B2
          Completions
                   Requirement: A14 | A15 | A16
                   Status: Completed
```

如果 A14、A15、A16 三个可分配单元中的一个完成了或通过了，块 8 就认为完成了。

示例 9： 第 3 级

```
Course.Structure
     Block
          SystemIdentifier: B21
          Completions
                   Requirements: 3*{A36, A37, A38, A39, A40}
                   Status: Completed
```

这告诉 CMI 系统，当这 5 个可分配单元中有三个完成或通过时块就完成了。比如说，有一个块有 5 个练习，课程设计者希望学生完成其中至少三个练习，就可以用这个逻辑表达式。

示例 10： 3A 级

```
Course.Structure
     Block
          SystemIdentifier: B13
          Completions
                 Requirement: A8=P | A9=P | A10=P | A11=P
                 Status: Incomplete
          Completions
                 Requirements: A8=P & A9=P & A10=P & A11=P
```

```
                    Status: Completed
```

注意，在这种情况下，块 13 永远不能达到完成状态，因为第一条语句总是在第二条语句之前为真，并且第一个为真的语句决定了课程元素的状态。

示例 11： 3A 级

```
Course.Structure
     Block
          SystemIdentifier: B13
          Completions
               Requirement: A8=P & A9=P & A10=P & A11=P
               Status: Completed
          Completions
               Requirement: A8=P | A9=P | A10=P | A11=P
               Status: Incomplete
```

这个例子修正了前一个例子的错误，现在，只要学生通过块中的一节课，这个块的状态就会从“Not attempted”变成“Incomplete”。当学生通过了所有的课，块的状态就会变成“Completed”。当第一条语句为 True, CMI 系统就永远不会根据第二条语句重新判定块状态。

示例 12： 3A 级

```
Course.Structure.Block
     AssignableUnit
          SystemIdentifier: A14
          Completions
                Requirement: A3=Fail
                Status: Fail
```

如果第 3 节课（可能是预测验）没通过(Failed)，第 14 节课就认为是没通过(Failed)。

7.5.8 结构考虑

这一节有很多例子，可以说明这一章讨论的许多原则。

示例 1

五节课必须按顺序从第一课学到第五课，学生只能在学完一节课后才能学习下一节课。

```
Course
     Structure
               AssignableUnit
                    SystemIdentifier: A1
               AssignableUnit
                    SystemIdentifier: A2
```

```
                    Prerequisite: A1
               AssignableUnit
                    SystemIdentifier: A3
                    Prerequisite: A2
               AssignableUnit
                    SystemIdentifier: A4
                    Prerequisite: A3
               AssignableUnit
                    SystemIdentifier: A5
                    Prerequisite: A4
```

示例 2

这五节课同属于一门课，第一课必须先学，第二、三、四或五课学习次序随意。

```
Course
    Structure
        Block
              Identifier: B0
              AssignableUnit
                   SystemIdentifier: A1
              AssignableUnit
                   SystemIdentifier: A2
                   Prerequisite: A1
              AssignableUnit
                   SystemIdentifier: A3
                   Prerequisite : A1
              AssignableUnit
                   SystemIdentifier: A4
                   Prerequisite: A1
              AssignableUnit
                   SystemIdentifier: A5
                   Prerequisite: A1
```

第 8 章 评价数据

课评价数据包含在几个类别之中。与通过 CBT 给 CMI 的通信文件所传递给 CMI 的信息相比，课评价数据包含更详细的关于学生表现的数据。

课评价数据通常存放在文件中，分析工具可以读取，至于文件是由 CMI 系统创建还是由单个可分配单元创建，依具体实现而定。因为评价数据完全可以脱离 CMI 和可分配单元之间的交流数据单独处理，这两类数据之间有很多冗余。

有了课评价数据，分析工具和 CMI 系统就可以把多节课的信息、同一节课的不同用法的信息，以及多个学生的信息汇聚在一起。

对信息的分析不是“规范”的所要讨论的内容。这里只谈原始数据，以及这些数据如何被分析工具所获取。

8.1 评价数据

下面以列表的形式给出“规范”所定义的课程描述数据，具体的数据元素的解释信息请参见“规范”文本。下面表格中的所有数据元素都是可选项。

表 21 评价数据

名称	中文名称	定义	列表	数据类型
Course ID	课程编号	由开发者提供的唯一的文字或数字标识	S	CMIIdentifier
Student ID	学生编号	代表 CMI 系统中单个用户的唯一的字母数字代码/标识符	S	CMIIdentifier
Lesson ID	课编号	由开发者提供的字母数字标识	S	CMIIdentifier
Date	日期	数据创建日期	S	CMIDate
Comments	注释	来自学生的自由形式的反馈	*	-
|--Time	时间	标识添加注释的时间	S	CMITime
|--Location	位置	标识在课的什么位置添加注释	S	CMIString255
|--Content	内容	学生注释内容的记录	S	CMIString4096
Interactions	交互	从学生到计算机的可检验、可记录的一组输入	*	-
|--Identifier	交互标识符	由课开发者创建的唯一的字母数字标识	S	CMIIdentifier
|--Objective IDs	教学目标标识	标识与交互相关的教学目标	*	CMIIdentifier
|--Time	交互时间	表明何时与学生交互	S	CMITime
|--Type	交互类型	表明哪种类别的交互被记录	S	CMIVocabulary
|--Responses	期望回答	在交互中期望学生给出的回答	*	--
|--|--Description	回答描述	在交互中学生可能回答的定义	S	CMIFeedback

名称	中文名称	定义	列表	数据类型
|--|--Value	分值	系统如何判定所描述的响应	S	CMIVocabulary
|--Weighting	权重	一个交互与另一个交互相比相对重要程度的比数	S	CMIDecimal
|--Student Response	学生回答	计算机可以测定的学生在一次交互中的行为描述	S	CMIFeedback
|--Result	评判	对学生反应的评价	S	CMIVocabulary
|--Latency	反应时间	从内容呈现到学生反应完成的时间	S	CMITimespan
Objectives	教学目标	关于学生对课目标完成情况的信息	*	-
|--Identifier	教学目标编号	由课开发者定义的唯一字母数字标签	S	CMIString255
|--Time	开始时间	表明学生何时开始目标学习	S	CMITime
|--Score	分数	纪录学生在学习教学目标有关内容的表现	S	--
|--|--Raw	原分数	学生在课上的得分	S	CMIDecimal
|--|--Maximum	最高分	学生得到的最高分或总分	S	CMIDecimal
|--|--Minimum	最低分	学生得到的最低分	S	CMIDecimal
|--Status	状态	用字母表示的一个教学目标的学习情况	S	CMIVocabulary
|--Mastery Time	掌握时间	花在这个教学目标上的总时间	S	CMITimespan
Paths	路径	学生在学习此节课时所历经的事件	*	-
|--Location ID	位置标识	学生处于课中的位置	S	CMIString255
|-- Time	进入时间	学生何时进入该学习元素	S	CMITime
|--Status	离开状态	学生离开学习元素时的表现纪录	S	CMIVocabulary
|--Why Left	离开原因	学生离开某个课元素的原因	S	CMIVocabulary
|--Time in Element	学习时间	学生学习该元素的时间	S	CMITimespan

8.2 交互示例

这个例子用来说明如何利用交互类中的数据元素标识学生在一节课的行为。

学生正处于一个测验当中，问题 3 是有 5 个答案的多选题。选中答案 B，得全分，选答案 C，却只能得一半分，其余答案都是错误的。在回答该问题时，学生选择了 C。

问题 4 是一个匹配题，有两列，每列有 4 项。第一列的每一项必须与第二列合适的选项

相匹配。因为这类问题要比多选题难，比如难 3 倍，所以它的权重是 3，多选题的权重为 1。

问题 5 是另一道多选题，有两个正确答案。学生若只选择了其中一个可得一半分，若两个正确答案都选择了则得全分。

```
Interactions.3
   Identifier: EngStart1
   ObjectiveID.1: Eng37
   ObjectiveID.2: Eng13
   Time: 14:23:00
   Type: multiple choice
   Response.1
      Description: B
      Value: Correct
   Response.2
      Description: C
      Value: .5
   Weighting: 2
   StudentResponse: C
   Result: 1
   Latency: 00:00:52
Interactions.4
   Identifier: SysPrep
   ObjectiveID: Eng17
   Time: 14:24:00
   Type: matching
   Response
      Description: {1.c,2.d,3.a,4.b}
      Value: Correct
   Weighting: 3.0
   StudentResponse: {1.c,2.b,3.a,4.d}
   Result: 0
   Latency: 00:01:12
Interactions.5
   Identifier: EngStart4
   ObjectiveID: Eng17
   Time: 14:26:00
   Type: multiple choice
   Response.1
      Description: a,b
      Value: .5
   Response.2
      Description: {a,b}
      Value: 1
```

```
    Weighting: 1.0
    StudentResponse: b
    Result: .5
    Latency: 00:01:03
```

8.3 路径实例

这个实例用来说明路径类中的数据元素如何标识学生课程活动的顺序。确定学生课程进度的顺序需要标识一系列的位置。每一位置都有一个标识符（位置标识）。位置标识的顺序反映了学生学习经历的顺序。

例如：假设有一节课有 6 个教学部分，每部分位置标识分别为 A, B, C, D, E, 和 F。学生们学习的路径这样表示：从点 1 到点 2，回到 3，以此类推。

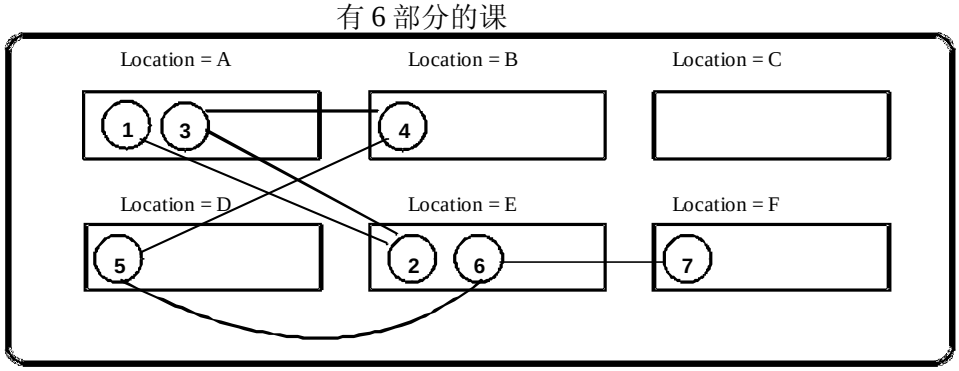

图 23 课程路径示例

图 23 所示的教学进程可以用下列数据元素定义路径。

```
Paths.1
     Location ID: A
     Time: 14:10:31
     Status: passed
     Why Left: student_selected
     Time in Element: 00:00:24
Paths.2
     Location ID: E
     Time: 14:10:55
     Status: passed
     Why Left: student_selected
     Time in Element: 00:01:06
Paths.3
     Location ID: A
     Time: 14:12:01
     Status: incomplete
     Why Left: lesson_directed
     Time in Element: 00:02:24
```

```
Paths.4
     Location ID: B
     Time: 14:13:25
     Status: passed
     Why Left: student_selected
     Time in Element: 00:00:54
Paths.5
     Location ID: D
     Time: 14:14:19
     Status: passed
     Why Left: lesson_directed
     Time in Element: 00:02:40
Paths.6
     Location ID: E
     Time: 14:16:59
     Status: passed
     Why Left: student_selected
     Time in Element: 00:03:03
Paths.7
     Location ID: F
     Time: 14:20:02
     Status: passed
     Why Left: exit
     Time in Element: 00:02:12
```

8.4 纲要

这里有一个模式可能对很多课程开发人员都有用，因为对课评价数据没有强制要求，所以不存在课评价数据的强制纲要。

```
CourseID: CMIString255
StudentID: CMIIdentifier
LessonID: CMIString255
Date: CMIDate
Comments.1
    Location: CMIString255
    Content: CMIString4096
Comments.2
    Location: CMIString255
    Content: CMIString4096
Interactions.1
    Identifier: CMIString255
```

```
    Type: CMIVocabulary
    Response.1
        Description: CMIFeedback
        Value: CMIVocabulary
    Response.2
        Description: CMIFeedback
        Value: CMIVocabulary
    Weighting: CMIDecimal
    StudentResponse: CMIFeedback
Interactions.2
    Identifier: CMIString255
    Type: CMIVocabulary
    Response.1
        Description: CMIFeedback
    Response.2
        Description: CMIFeedback
    StudentResponse: CMIFeedback
Objectives.1
    Identifier: CMIString255
    Score: CMIDecimal
    Status: CMIVocabulary
Objectives.2
    Identifier: CMIString255
    Score: CMIDecimal
    Status: CMIVocabulary
```