第 33 卷 第 7 期 计 算 机 工 程 2007 年 4 月
Vol.33 No.7 Computer Engineering April 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)07—0282—04 文献标识码：A 中图分类号：TP391

基于 Web 的网络旅游服务系统的形式化 B 开发

李信本
（浙江万里学院计算机与信息学院，宁波 315100）

摘 要：形式化 B 方法支持从规格说明到代码生成的全部软件开发过程。结合网络旅游服务系统模型，讨论了形式化 B 方法的具体运用，在分析服务器端和客户端状态表示的基础上，给出了该系统的抽象机模型及其精化过程。
关键词：形式化方法；B 方法；Web 技术；抽象机

Development of Web-based Travel Agency System Using B Method

LI Xinben
(School of Computer and Information, Zhejiang Wanli University, Ningbo 315100)

【Abstract】 B method is one of the formal methods, which supports the whole process of software development from the specification to code generation. This paper applies the B method to Web-based travel agency system, based on the analysis of the state representation in server and client side the formal abstract machine as well as its refinements is given.
【Key words】 Formal method; B method; Web technology; Abstract machine

随着经济全球化和社会信息化步伐的加快，基于 Web 技术的全球电子商务应用已成为信息化社会的主要商务模式。由于基于 Web 的系统是可用 Web 浏览器访问的分布式系统，其复杂性和开发难度随着系统规模的扩大而急剧增长，严格的系统化开发方法在开发此类系统中就显得日益重要。

形式化 B 方法作为一种“基于模型”的软件构造方法，其开发过程包括需求分析、规格说明、精化和实现等步骤，并在整个开发过程中通过正确性验证，保证了软件产品的高可靠性、可移植性和可维护性，提高了软件生产率。本文研究将形式化 B 方法应用于网络旅游服务系统的设计实现方法，重点讨论 Web 用户接口、核心业务处理、数据存储等的规格说明和精化。

1 形式化 B 方法概述

形式化方法是建立在严格数学基础上的软件开发方法。在软件开发过程中，从需求分析、规格说明、设计、编程、系统集成、测试、文档生成直至维护各阶段，凡是采用严格的数学语言、具有精确的数学语义的方法称为形式化方法[3]。B是一种基于模型的软件构造方法，类似于VDM和Z[1]。它是建立在命题演算、谓词演算、等式和有序对、基本和派生的集合构造、二元关系、函数、广义并和交等数学理论之上的形式系统，其中还包括了广义代换、精化、软件的层次体系结构等理论。

B 方法根据功能需求进行正确性验证，保证软件产品的高可靠性和可维护性。该方法使用 AMM(Abstract Machine Notation)作为软件开发过程中的规格说明、设计及实现语言，包括有赋值和条件等程序设计结构以及前条件、约束选择、卫、多重赋值等广义代换的特征。广义代换语言(Generalized Substitution Language, GSL)提供 AMN 的形式语义，标识 AM 的操作，通过在 GSL 中引入序列和循环结构，并重写 AMN 中的操作以便得到更接近于可执行代码的控制结构（最终的命令式程序代码由代码生成器自动转换得到）。

使用 B 方法开发时，首先需要为软件系统建模，用 B 的表示来描述系统变量的主要状态、属性及其在操作中的转变，为系统书写规格说明，并在此基础上逐步求精，直到产生一个软件系统的完整实现。此方法通过一组计算机辅助工具支持全过程的开发工作。B-Toolkit 包含自动及交互的理论证明工具及一套软件开发工具：AMN 类型检验器(Type Checker)，证明义务生成器，规格说明和代码生成器。所有开发工具都由一个公共平台 B-Tool 支持。最后通过模式匹配和规则重写机制，对形式化对象生成程序。B 语言中结构化的机制与其它面向对象方法一样，增强了信息隐藏和数据封装，严密的部件接口控制确保了大型软件开发中各个部件的独立开发。

“抽象机”(Abstract Machine)是 B 的基本封装机制，具有机构化特征。B 方法涉及到的一些主要思想包括：

(1)数据规约：AM 中的数据通过一组数学概念（如集合、关系、函数、序列、树等）说明，这些数据通过一些所谓的不变式(Invariant)的确定条件来规定其必须遵守的静态规则。

(2)操作规约：通过不包含顺序性和循环的非执行伪码表示，每一操作被描述为一个前置条件(pre-condition)和一个原子动作(atomic-action)，前置条件是此操作的被激活的必要条件，原子动作通过广义代换方式来形式化表示。所有的原子动作符合不确定性选择策略。

(3)精化技术：AM 的最初模型（即它的规格说明）可以被精化为对应的可执行模块（即它的实现）。精化过程是逐步的，并非一次完成，前后两次精化过程之间伴随有正确性证明过程。在每一次精化过程中，AM 要重新全部构造，尽管相应的伪代码有某些修改，但对用户而言，它保持相同的操

基金项目：浙江省教育厅资助项目(20031272)
作者简介：李信本(1960－)，男，硕士、高工，主研方向：软件工程，形式化方法
收稿日期：2006-05-19 **E-mail：**xb8371@zwu.edu.cn

作。在中间的精化步骤里，通过某些数据或操作的变换来实现，实际上是一种混合构造过程，不再是纯数学模型，当然还不是真正意义上的程序设计模型。

(4)分层结构：将一个大而复杂的精化分解成更小的模块，然后通过递增式规约机制来实现系统的层次体系结构。

(5)部件库：在 B 方法中，预先提供一组已定义好的 AM（是一些典型数据结构的封装），用户在精化 AM 的过程中可直接使用它们。另外，部件库可扩充，提供了可重用机制。

(6)代码生成：AM 的最终精化通过代码生成器可以很方便地翻译成不同的命令式程序设计语言的程序。

2 网络旅游服务系统的功能需求

网络旅游服务系统是一个运行在旅行社 Web 服务器上的软件系统。游客通过互联网浏览器访问该系统，生成并提交期望的旅游项目（路线）服务请求。系统根据游客提交的服务请求，处理相关的旅行安排（包括预定机票、客房等）并将结果反馈给游客，在游客确认无误后，即可进行后续的工作。网络旅游服务系统完成游客服务请求还需借助于一些别的系统如机票预售系统、客房预定系统等，这些系统间通过互联网来通信，各系统之间呈现明显的分布式特征，如图 1。

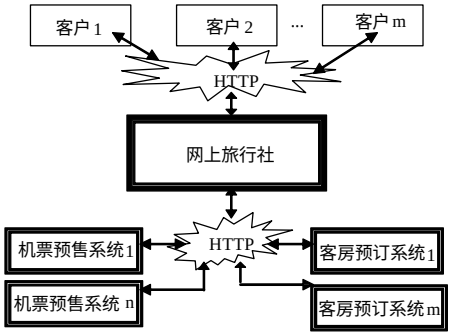

图 1 网上旅行社系统结构

网络旅游服务系统的软件体系结构是一个多层的网络应用结构，包括客户界面层、业务处理层、数据存储层。通常，上述三层程序分别安装于客户机、应用服务器和数据库服务器上，而实际运行中，客户界面层的程序是从应用服务器上下载的。因此，客户界面层和应用服务器是系统实现的关键。本文将着重对 B 语言设计的应用服务器层的设计与实现进行分析。

3 网络旅游服务系统的 B 规格说明

在基于 Web 的应用中，业务逻辑作为系统的核心功能是由服务器来处理的，大多数应用（如网上银行、电子商务）需要维护服务器与其客户间的通信会话，并监督每个客户的状态和活动。但 Web 浏览器和 Web 服务器之间的通信在 HTTP 协议下是无状态的，且 HTTP 协议没有会话控制功能。因此，如何维持服务器与每个客户交互的信息是设计工作的首要决策。对于这种服务器为中心的体系结构，自然的选择就是在服务器端保持并控制会话，本文从服务器端的设计开始，先看其抽象机，网络旅游服务系统的 B 抽象机模型如下所示：

```
MACHINE TravelAgency
SETS
  SESSION;
  STATE={ fresh, booking, unbooking, service_selct, option_ret,
    choice_made,, Signed_in, certified, valid, invalid,
    booking_ret, unbooked_sel };
DEFINITIONS
  FreshSESSION≡SESSION – session;
VARIABLES
  session, session_state;
INVARIANT
  session ≤SESSION ∧
  session_state∈session → STATE ∧ …
INITIALISATION
  session := ϕ|| session_state := ϕ || …
OPERATIONS
  StartNewSession ≡
Any sid WHERE sid∈freshSESSION THEN
  session := session ∪ {sid} ||
  session_state(sid) := fresh;
END;
  SelectService ≡
Any sid WHERE sid∈session ∧
 session_state(sid) := fresh THEN
  SELECT (……) THEN
    session_state(sid) := booking;
  WHEN (……) THEN
    session_state(sid) := unbooking;
  WHEN (……) THEN
    session_state(sid) := signed_in;
  END ||…
END;
  FlightRequest ≡
Any sid WHERE sid∈session ∧
 session_state(sid) := booking THEN
  session_state(sid) := service_selct;
END;
```

服务器利用一个会话的状态变量来辨识一个用户、处理客户提供的输入数据、确定用户权限或要提供给用户的服务类型。并且，根据用户已提供的信息，服务器能够设置状态变量以决定下一个可能的执行路径。

定义两个集合分别表示会话状态和会话标识，再定义一个从会话标识到会话状态的映射函数，这样，就能在服务器端标识并管理每个会话。每个会话有一个会话标识“ sid ”，该值可用做服务器端存取会话信息的下标。一旦一个新客户与服务器建立了连接，则立即为其分配一个新“ sid ”，客户在后续交互中就可使用该 sid。

网络旅游服务系统的 B 抽象机模型中，集合 STATE 表示一个会话的所有可能状态，集合 SESSION 用做会话标识的类型引用。变量 session-state 映射每一个客户会话到其相关状态，变量 session 表示当前所有活动的会话所构成的集合。操作 StartNewSession 为系统生成新会话构造模型，该操作分配一个空的会话标识给新生成的会话并为其设置必要的环境变量。会话状态变量的任何改变引发一个操作，而且执行一个操作会导致状态变量的某些变化，例如，当会话的状态是 fresh 时，触发操作 SelectionService 开始执行，并且执行该操作将改变相关会话的状态成为 booking、unbooking 或 signed-in 之一。当客户选择一个可用的旅游服务项目时，操作 SelectService 就是系统处理客户服务请求的模型，而该操作的执行又会触发预订航班机票的操作 FlightRequest 和预订宾馆客房的操作 RoomRequestk 开始执行，从而完成客户请求的旅游服务安排。由于相似性，这里略去了 RoomRequest 操作的规格说明。

4 B 抽象机的精化

网络旅游服务系统的 B 抽象机模型描述的抽象机尚未考虑客户端如何维持与服务器交互的会话状态，也未考虑服务器与其它系统（如航班订票）的交互会话，这也是精化过程的任务。

4.1 客户端的状态管理

首先，系统如何在客户端维持会话状态信息并完成不同客户与 Web 服务器之间的交互，Web 客户层接受用户的输入数据、完成类型检查和简单的数据校验，然后提交这些信息供服务器处理。由于各个客户以异步方式独立且并行地与系统交互，系统无法控制客户的动作及动作发生的时间。尽管浏览器及其支持机制无须支持状态处理，但在服务器和客户操作之间仍必须维持某些协调机制和状态传递任务。

使用消息机制可实现客户端的会话状态表示，即在客户端，消息机制可看为是一种隐式状态表示，其优点是避免了在客户之间及客户与服务器间共享状态变量。每个消息映射到一个会话标识，该标识将消息与一个特定的客户会话相关联。

随着对服务器抽象机的操作的第 1 次精化，给出了客户端的一些操作，也补充了一些系统必须的变量信息。本文用注释来区分服务器端的操作和新引入的客户端的操作，服务器端操作使用显式的状态变量来表示状态，而客户端操作使用隐式的基于消息的方法表示状态及协调服务器的操作。会话标识 sid 在客户与服务器之间传递状态信息起着核心作用。然而，有时客户请求了一个新的会话，但还没有获得一个会话标识。此时客户应使用一个临时的辨识机制（如 IP 地址）或其它类似机制。抽象机的第 1 次精化中的 Client_ReqSession 操作描述了这种情况，本文用一个名为 handle 的新变量作为临时下标来表示客户的新会话请求，当服务器在 StartNewSession 操作中已处理了该请求后，服务器为这个特定客户分配一个新会话标识，并通过将该新会话标识存入 New_Client 消息缓冲区来回答客户。在 Get_SessionID 操作中，客户接收到这个已分配的 sid，然后就一直使用该会话标识来完成与服务器的后续通信。例如，在 picService 操作中，使用了一个名为 reqService_buf 的消息缓冲区，该缓冲区定义为 session 到 REQUEST 的映射，用以执行客户对服务器的请求。其中 REQUEST 是一个枚举集合，用来表示客户请求的服务。抽象机的第 1 次精化如下：

```
SETS
   HANDLE;
   REQUEST；
   RESPONSE;
DEFINITIONS
   freshHANGLE≡HANDLE – dom(new_client);
VARIABLES
  handle, new_handle, token, req, resp
INVARIANT
  /*Client variablew */
   new_handle ≤HANDLE ∧
   new_client∈HANDLE  +> SESSION ∧
   token ≤SESSION ∧
   fresh_session ≤SESSION ∧
   reqservice_buf ∈ SESSION  +>  REQUEST
   resp_buf ∈ SESSION  +>  RESPONSE
INITIALISATION
  handle := ϕ|| new_handle := ϕ||token := ϕ||…
OPERATIONS
  Client_ReqSession ≡ /* client Operation */
    ANY handle WHERE handle∈freshHANDLE THEN
        new_handle := new_handle ∪ {handle}
    END;
  StartNewSession ≡ /* server Operation */
    ANY sid, handle WHERE sid∈freshSESSION ∧
                handle ∈ new_handle  THEN
      session := session∪{sid} || session_state(sid) := fresh ||
      new_client(handle) := sid ||
      new_handle := new_handle －{handle}
    END;
   Get_SessionID ≡              /* client Operation */
    ANY sid WHERE sid∈SESSION ∧
              sid∈ran(new_client)  THEN
        token := token ∪ {sid} ||
        fresh_session := fresh_session ∪ {sid} ||
        new_client := new_client ⊙ {sid}||
    END;
   PicService ≡  /* client Operation */
    ANY sid, req WHERE sid∈fresh_session ∧
            Sid /∈dom(reqservice_buf) ∧
            req ∈ REQUEST ∧ req ≠none THEN
        Reqservice_buf(sid) := req ||
        fresh_session := fresh_session – {sid}
    END;
  SelectService ≡ /* server Operation */
    ANY sid, req WHERE sid∈session∧req∈REQUEST
          ∧Resp∈RESPONSE
          ∧sid∈dom(reqservice_buf) THEN
        session_request(sid) := req ||
        reqservice_buf := {sid} ◎ reqservice_buf ||
        resp_buf(sid) := resp
    END;
Submit_Service_Dtail  ≡ /* client Operation */
    ANY sid, resp WHERE sid∈dom(resp_buf) ∧
      resp∈RESPONSE ∧ resp_buf(sid) := resp THEN
      resp_buf(sid) := {sid} ◎ resp_buf(sid)
    END;
```

4.2 服务器与其它系统之间的通信

不像客户与服务器之间的通信，服务器与其它系统的服务器之间的通信会话涉及的双方都在提供某些服务。考虑到各个服务器互相独立这一事实，为服务器间的交互建立模型的任何方法应尽可能减少这些子系统间的耦合。采用基于消息的方法是一个不错的选择且符合公共 Web 服务技术，因为消息既能在各系统的服务器之间交换数据，也能交换状态信息。这些消息被定义为从一个会话标识到请求信息的映射，其中 reqflight_buf 消息缓冲区用来传输服务器关于航班信息的查询请求到各航空公司售票系统，航空公司售票系统利用 respflight_buf 消息缓冲区送回该公司的航班信息给服务器。相应地，网络旅游服务系统的 B 抽象机模型中的操作 Flight Request 将被精化为服务器查询航班信息的操作 Request_Flight 以及航空公司响应查询请求的操作 Resp_FlightReqs。类似地，服务器与各个客房预定系统的交互也需定义相应的消息缓冲区和传递消息的操作，这里从略。

4.3 分布式数据库的精化

通常，各个航空售票系统是用数据库来维持预售票信息

及其数据的持久性的，不同航空公司的售票信息数据库一般分布在不同场地的服务器上。建立分布式数据库的形式化模型是一个较难处理的问题，因为这些数据库为多个用户所共享，查询时可用的机票在真正预定时也许已经售出。因而，具体的订机票操作在分布式环境下应从下述两个方面考虑。

(1)前述的查询航班信息的操作FlightRequest应进一步精化成一个分布式查询，并以广播方式发送给所有航空售票系统以检索信息。由于操作 Resp_FlightReqs 仅是各航空售票系统返回的该公司的可售机票信息，实际的订票方案应全面评估所有航空公司返回的信息后来制订，因此，新增加的操作 Retrieve_FlightOptions 专门用来收集所有航空公司返回的可售机票信息。

(2)在确定要预订某个特定航空公司的某航班机票后，由于多用户共享的缘故，该航班机票也许已售出，因此，实际的订票过程可精化成两个操作（分别表示售票过程的两个不同阶段）：操作 Agency_flight_booking 描述了当请求的航班机票在特定的航空公司仍然可用时，发生在该航空公司的售票服务器上的订票过程；操作 Flight_Booking 描述了当系统收到该航空公司售票系统发来的订票成功的消息时，系统将订票结果添加到自己的数据库的过程。每个航空售票系统服务器的订票数据库仅存放该公司已提供的订票服务，旅行社系统的订票数据库存放所有用户已订好的服务。

上述的精化结果如下所示。其中抽象数据类型 FLIGHT_REQUEST 和 FLIGHT_DETAIL 分别表示查询航班信息的记录的集合和航空公司可供出售的航班机票的集合，常量函数 MatchFlight 根据航班查询信息和当前可供出售的机票信息返回满足要求的可预订机票信息。

分布式数据库的精化结果：

```
CONSTANTS
  MatchFlight
PROPERTITIES
  MatchFlight∈FLIGHT_REQUEST ×
       P(FLIGHT_DETAIL) → (FLIGHT_DETAIL)
INVARIANT
  Flight_db∈FLIGHT_AGENCY /→P(FLIGHT_DETAIL)
OPERATIONS
  Request_Flight≡          /* Server Operation */
    ANY sid, fr WHERE sid∈SESSION∧
        fr∈FLIGHT_REQUEST   THEN
        Reqflight_buf := λfa.(fa∈FLIGHT_AGENCY|
           reqflight_buf(fa)∪{sid ├>fr}
    END;
  Resp_FlightReqs ≡ /* Flight Agency Server Operation */
    ANY sid, fa, fr   WHERE sid∈session∧
          fa∈FLIGHT_AGENCY∧
          fr∈FLIGHT_REQUEST   THEN
        ANY xx WHERE xx∈P(FLIGHT_DETAIL)∧
             xx≤MatchFlight(fr ├>flight_db1(fa))   THEN
          Respflight_buf(sid):=respflight_buf(sid)∪
               {fa ├>sid }
        END
    END;
  Retrieve_FlightOptions≡   /* Server Operation */
    ANY sid WHERE sid∈session THEN
        Flight_options(sid):=∪fa.(fa∈FLIGHT_AGENCY∧
          fa∈dom(respflight_buf(sid))| respflight_buf(sid)(fa))
    END;
Agency_Flight_Booking ≡   /* Flight Agency Server */
   ANY fa, sid, fd WHERE fa∈FLIGHT_AGENCY∧
        sid∈SESSION∧fd∈FLIGHT_ DETAIL THEN
      SELECT fd∈flight_db1(fa) THEN
        ANY fdb WHERE fdb∈P(FLIGHT_DETAIL)∧
            fdb≤flight_db1(fa) THEN
          flight_db1(fa) := fdb
        END ‖
        fa_booking(fa) := fa_booking(fa) ∪
                {fd ├>session_user(sid)} ‖
        flightbookingresp(sid) := success
        WHEN fd /∈flight_db1(fa) THEN
          flightbookingresp(sid) := failed
        END
    END;
Flight_Booking≡          /* Server Operation */
    ANY sid, fa, fd WHERE sid∈session ∧
        fd∈FLIGHT_ DETAIL ∧
        fa∈FLIGHT_AGENCY   THEN
      SELECT sid ├>success∈flightbookingresp THEN
        taf_booking := taf_booking∪
              { session_user(sid) ├> fd ├> fa }
          Suc_session:= Suc_session∪{sid}
      WHEN sid ├>failed∈flightbookingresp THEN
          Selectflight_buf(fa):=selectflight_buf(fa) –
                {sid ├> fd} ‖
          unsuc_session:= unsuc_session∪{sid}
      END
    END;
```

5 实现与讨论

经过前述的 3 个精化步骤，网络旅游服务系统的 B 抽象机模型的抽象机已经接近于系统的实现。然而，完整的系统实现还需继续精化过程，尤其是关于分布式数据库的精化还要结合其它系统具体的数据库数据模式来分析讨论，另外，完整的 B 开发还包括规格说明的推理和证明，限于篇幅，将另文给出。

本文结合网络旅游服务系统讨论了形式化 B 方法的具体运用，在分析服务器端和客户端状态表示的基础上，给出了该系统的抽象机模型及其精化过程。不可否认，要全面掌握 B 方法的开发并非易事。然而，B 方法中的精化和实现是很多形式化开发方法所没有的，它在规格说明的基础上可直接生成可执行系统，并在整个开发过程中通过正确性验证，保证了软件产品的高可靠性、可移植性和可维护性，从而可有效提高软件的生产率。

参考文献

1 Abrial J R. The B-Book:Assigning Programs to Meanings[D]. Cambridge University, 1996.

2 肖美华, 薛锦云. 形式化方法B及其程序规约机理[J]. 计算机工程, 2004, 30(8).

3 塔维娜, 何积丰. 基于形式化方法的需求分析[J]. 计算机工程, 2003, 29(18): 107-108.

4 李　莉, 缪淮扣. 一个基于 Web 技术的网络银行系统的 B 方法设计与实现[J]. 计算机工程, 2001, 27(10): 65-66.