The ChocAn Simulator

Design Document

Written by Group #11

Ricardo Cruz

Jonathan Bral

Gilbert Grundy

Andrew Bonham

Table of Contents

1 Introduction

This design document describes the design for ChocAn's data processing software. It will outline design considerations, system overview, system architecture, and detailed system design.

1.1 Purpose and Scope

This document will be used to both validate software design decisions and to guide the software development team in the software implementation phase. This will be achieved by setting out important design considerations for the ChocAn data processing software, a broad overview of the system design, and a more detailed description of the software architecture and design.

1.2 Target Audience

The document is intended to be read by the senior management team of ChocAn and the software development team of Just Four Guys Programming Co..

1.3 Terms and Definitions

- BST - Binary Search Tree
- ChocAn - Chocoholics Anonymous, an organization dedicated to helping people addicted to chocolate.
- ChocAn Data Center - Database which holds member information.
- Electronic Funds Transfer (EFT) - Transfers currency from one account to another.
- HIPAA - Health Insurance Portability and Accountability Act, a bill passed by Congress in 1996, which mandates industry-wide standards for healthcare information.
- Interactive Mode - A mode which provides the user the ability to add, remove and update both members and providers.
- low-level software permissions.
- LLL - Linear Linked List
- Manager - Administrative entity and user with admin. Software permissions
- Manager Terminal - a computer terminal accessible only to managers
- Member - A client of ChocAn who pay a monthly fee to have access to treatments provided by their providers.
- Object - An entity that contains certain attributes that relate to the context of its use.
- Operator - An employee at the ChocAn Data Center
- PHI - Protected Health Information
- Product Owner - Development team's point of contact with the customer, ChocAn.
- Provider - Health care professional who provides services to clients of ChocAn, with
- Provider Directory - A list of all providers stored on disk.
- Provider Reports - A weekly summary sent to providers which includes total fees
- Providers Terminal - A specially designed computer terminal, similar to a credit card machine in a shop.
- Service - counseling, healthcare, and lifestyle advice.

2 Design Considerations

For the ChocAn data processing software, there are several constraints and dependencies to consider for the design of the software. They are outlined here in this section, as well as the methodology used for the software design process.

2.1 Constraints and Dependencies

- HIPAA.
- Reading and writing data to and from files.
- Storing files and accessing.
- Encryption of information (ID Numbers, medical records).
- Accounting procedures handled by third party (Acme).
- Third party contractor developing communication software.
- Product deadline.
- Do not have access to actual hardware.

2.2 Methodology

The methodology implemented for the development of the ChocAn software, is the Waterfall development process. This product is being sequentially developed, starting from the requirements stage, into the design stage, and then into implementation, verification, and validation stages. If at any stage of development, an unforseen requirement arises, the team will go back through both the requirements and design stages, to account for these changes that need to be made in the implementation (or later) stage. By approaching the products development in this way, allows us to deliver a properly functioning product with the first release, eliminating the need to release further iterations of the software, or any prototyping.

The implementation and design of the software will be object orientated. By using an object orientated design we are able to break the responsibilities of the data structure and data management into smaller more specific jobs. This will allow for an easier understanding of what each class' role is in the implementation and should allow for an easier experience in editing and maintaining the software. The programming language that will be used is going to be the object orientated language, C++, giving the developers a lot of power over the management of memory.

3 System Overview

The ChocAn system overview is represented in figure 3.1. The blue highlighted section represents the part of the ChocAn system for which the Just Four Guys Programming Co. are responsible for developing.

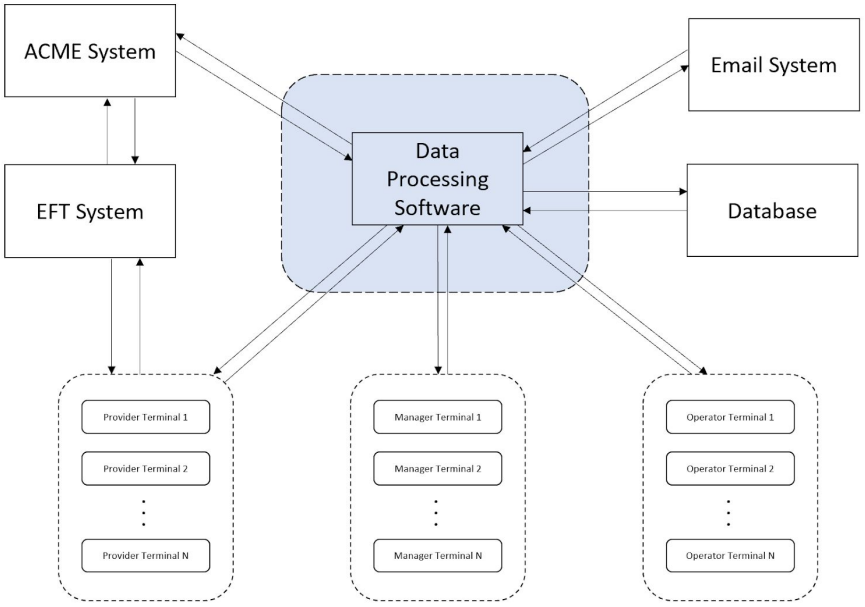


Figure 3.1

The Data Processing Software that we are designing provides the interface between the hardware provided by the terminals of providers, managers, and operators and the software of the database, email system and Acme system. The provider terminal will also have an EFT component, which will be the responsibility of Acme systems in implementing. Furthermore, Acme is responsible for financial procedures such as recording payments of membership fees, suspending members whose fees are overdue, and reinstating suspended members who have brought their accounts back into good standing.

The e-mail system will have the capability of sending a provider directory, member reports and provider reports; the responsibility for the sending of data is of the contractor for the communication software, while the Data Processing software will be responsible for building the reports and will stimulate these requests with screen output and writing to disk.

Provider terminals will need to access member, provider and service data to perform transactions. Secondly, operator terminals will be responsible for maintaining the database of members, providers and services. Additionally, the managers terminal can request a member report, provider report, and a summary report at any time. It is the responsibility of the data processing software to provide the interface between the terminals and the database for these functions. For this project the database will be stimulated with text files.

4 System Architecture

The architecture of the system will be built around several binary search tree data structures. One each for the Providers and Members of ChocAn, along with a binary search tree for the Services provided, as shown in figure 4.1. These will be built by reading in data from a text file, which is acting as our database, at runtime.

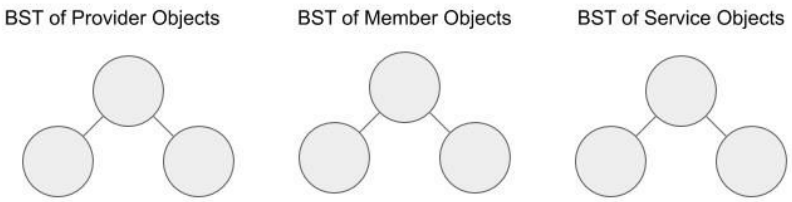


Figure 4.1

4.1 Classes

The class hierarchy of objects in the data processing software is displayed in figure 4.2. We have an ID class which acts as an Abstract Base Class (ABC) and binary tree node. The service class and person class are derived from ID class. The Provider and Member classes are then derived from the Person class and they contain a head pointer to the Record class, which acts like a node to a Linear Linked List (LLL). Finally, the Manager class is friends with the Provider and Member classes.

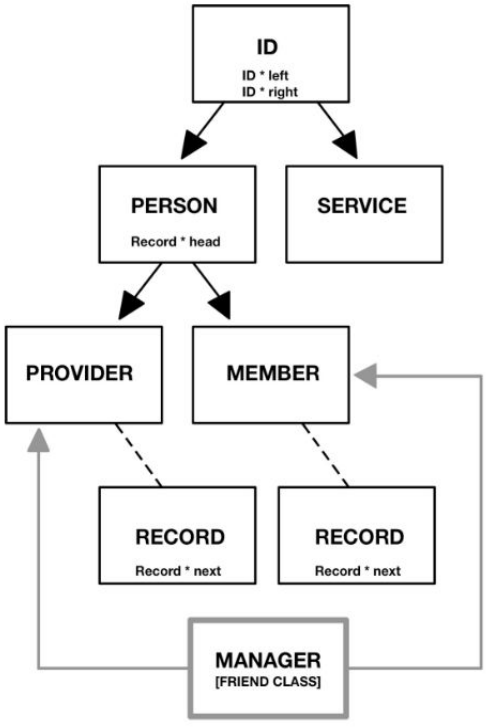


Figure 4.2

4.1.1 ID Class

The highest class in the hierarchy is the abstract base class. The abstract base class contains an ID Number and two pointers to the ID Class, a left and a right. The two pointers will be used for the binary search tree data structure which will be built from this class. Methods will provide the ability to traverse and search the tree.

4.1.2 Service Class

The Service class inherits the abstract base class ID. This class is for managing, displaying, and containing all of the Services provided by ChocAn. This class contains the service name, service code, and fee associated with the service.

4.1.3 Person Class

The Person class inherits the abstract base class. The Person class also contains several variables that the Provider and Member classes have in common, things like a person's name, street address, city, and zip code.

4.1.4 Member and Provider Classes

The Member and Provider classes inherit the Person class. There will be both specific methods and data members specific to each class. For example, Members will contain a float representing account balance and Providers will include a string which will represent role.

4.1.5 Record Class

The Record Class is owned by the Member and Provider Classes. This class will be used to create a Linear Linked List of data for the sales history used by the appropriate member/provider of ChocAn.

4.1.6 Manager Class

The Manager Class is a friend class to the Member and Provider classes, giving the manager class access to the data members inside the respective classes.

5 Detailed System Design

5.1 Object Oriented Programming (OOP) Hierarchy

Provider and Member objects are subclasses of a common parent class, that is the Person class, which has all common traits (data members) used by member and provider objects, including Name, Street address, City, State, Zip code, and a Record class pointer called head. The record class will have a Record class pointer called next.

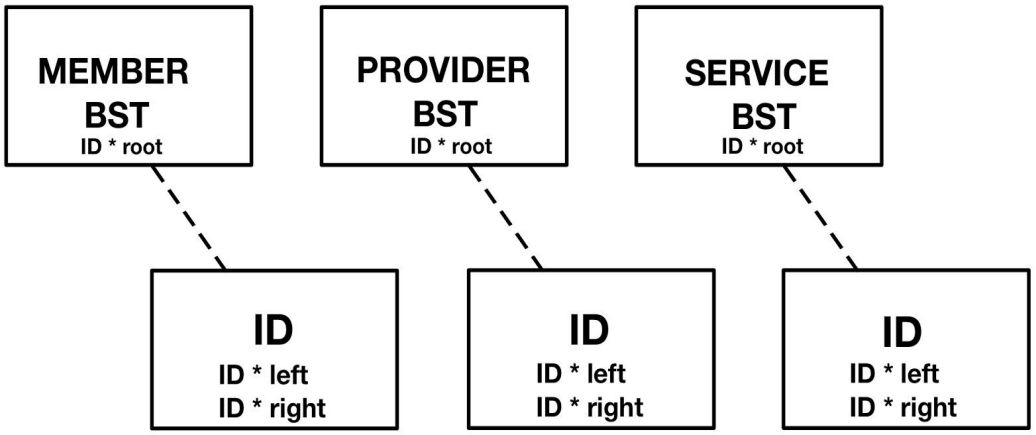
The base class of the hierarchy, Person, is derived from is an abstract base class with an ID data member along with two ID class pointers named left and right, respectively, which will be pushed up into this base class. Service is another subclass of this abstract base class and contains data members that represent the name of the service, service code, and the service fee.

5.2 Data Structures

A macro view of this design will reveal that there will be a LLL of records in each node of the Member BST and Provider BST.

5.2.1 Binary Search Trees (BST)

Each BST data structure will have an instance of the ID base class. Since the ID class contains both a left and right pointer instance of itself, each BST can populate several instances of the respective class type after an ID pointer has been dynamically cast. The Service BST will be alphabetically ordered and will serve as a Provider Directory accessible by the provider. Each service will be stored in a text file and read in to populate the Service BST. Members and Providers will similarly be stored in a text file and will also populate their respective trees.



5.2.2 Linear Linked Lists (LLL)

The OOP design choice enables the Provider and Member classes to be dynamically bound with each inheriting a pointer to the Record class. This will allow for the population of a Linear Linked List (LLL) representing a Provider's history of verification forms and another LLL that represents a member's history of services provided.

5.3 Methods

Methods for retrieval and sorting will be applied to the data structures. Retrieval will be used to obtain data from the Provider BST, Service BST, and Member BST. A sorting method will be necessary to give the data in each of the BST's in alphabetical order. Operators will also need some basic methods to manage the data, such as add, delete and edit.

5.3.1 Retrieval

A chocAn Manager will have the ability to access Provider and Member BSTs from within their manager terminals. As a user, the ability to retrieve a specific member's or provider's service history is made possible through methods belonging to the Provider BST and Member BST. A member class will use the returned data to access Member or Provider data members and their respective file directories on disk. A member class can also make use of methods such as display and retrieval on Record LLLs to aid in compiling weekly reports. These reports are then stored on disk in a way that they can be sent to Acme for accounts receivable services.

A provider will enter their provider number, and a method belonging to the Provider BST class will accept the number as an argument and traverse the tree until a match is found. If no match then the method will return -1 and an message on the terminal will display "Invalid ID Number". If successful, a copy of the Provider class containing the relevant data will be returned. The user will then enter the Member's ID number, a method belonging to the Member BST class will accept the member ID number as an argument, and the method will traverse the tree in search of a match. An error message will be displayed if there is no match or if a match is found and the user is not validated. However, if the member is valid, then a copy of the Member class containing the member data will be returned.

Once the user in the provider terminal has completed their service, the provider will once again validate the member through a method belonging to the Member BST. Once validated, the user will invoke the Provider Directory. A Service BST method will display all services in the BST tree. The user will enter the relevant service code which will be stored using a local variable in main. A method belonging to Service BST will accept the service code as an argument and use it to locate the service.

5.3.2 Sorting

Since the Service BST is sorted by ID number, we will need a sorting algorithm to give a list in alphabetical order. To do so, a temporary array the size of the tree will be created and loaded with services out of order. Then a private method will use a quick sort algorithm to sort the list into alphabetical order. The list will then display and be written out to disk. This method will return an integer indicating success or failure.

5.3.3 Add

Adding data to any of the the BST trees is quite trivial as you always add at a leaf. There will be a method that traverses the tree using comparing methods until reaches a NULL leaf to add a new member, service, or provider.

5.3.4 Delete

Removing data from the BST tree is slightly trickier as there is corresponding information from the provider tree and service tree. If a provider is removed it is important that the service tree is also traversed to remove corresponding services given by that provider. The method that removes a service provider should also then search the service tree and remove the corresponding services.

5.3.5 Edit

Given an ID key, an operator may edit a member, provider or service. To do so there will be a method which allows the operator to traverse the tree and once a match is found, edit the appropriate data members. If no match is found then the method will return 0.

5.4 Security

All data members pertaining to Provider, and Member’s personal information will be stored on disk in a text file format. ID numbers will be stored on disk as well; however, for security purposes, they will be hashed using the SHA-2 algorithm provided by the OpenSSL C++ Library.

5.5 Writing to Disk

Once the session is completed, a Provider method will accept the Member and Service copy that was returned in previous methods. This method will write a record with the relevant data to disk in the form of a text file along with also inserting a more detailed record into a record LLL. Similarly, a Member method will execute upon return and will accept the Provider and Service copy that was returned in previous methods. The method will also write a record to disk and insert a copy to the Member’s LLL.

The Record class will also hold strings which are addresses to text files in a weekly directory. This directory will have text files written into when services are input into the system by providers. These same text files will then be accessed when reports are run by methods in the manager class. To accomplish this system we will have a Provider method, which will add a Record to both LLL of the corresponding Member and Provider objects, which points to the added file. There will then be a seperate Manager method, which searches for these LLL to access the text files, when found it will then read in the text file and write out the appropriate report to a report directory.