OpenBazaar Redevelopment - Design Document

The Fair Traders
Daniel Mandel - mandeldr
Shandelle Murray - murras25
Connor Sheehan - sheehacg

November 6, 2015

Abstract

This documents outlines design for the OpenBazaar redevelopment project.

Table of Contents

Revision History

Revision Number	Revision Date	Description of Change	A
1	November 4th, 2015	Created Revision History	Dan
2	November 6th, 2015	Added to Introduction, added numbering, created tables	Shand

Table 1: Table to capture the history of the document

Introduction

0.1 Purpose

The purpose of this document is to describe the implementation of the OpenBazaar that was described in the Software Requirements Specification (SRS) document completed earlier this semester. It aims to outline a design that will meet all of the functional and non-functional requirements described in the SRS. It is also meant to be a template for creating the Module Interface Specification document, MIS, which will describe the modules in further detail.

The design principle being used to implement this project is the principle of information hiding which was first described by David Parnas (Parnas,1972). The idea behind this design strategy is that each module contains some secret, essentially hiding a design decision from the rest of the system. As a result of this method of modularization, aspects of the system that are likely to change are hidden within a module and, when changed, do not affect the rest of the modules. This is important for any software design as technology is constantly evolving and software often needs to be updated in order to remain relevant.

This document is intended for future developers and designers who wish to improve or better understand the design of the OpenBazaar. It is organized into sections of anticipated and unlikely changes to the design, a description of the module hierarchy, a decomposition of each module in the design, and traceability matrices demonstrating the connections between modules and requirements as well as modules and anticipated changes.

0.2 Scope

The purpose of this project is to design and implement OpenBazaar, a free, open market run through a peer-to-peer network that aims to replace centralized services such as eBay or Amazon by providing a means in which to participate in online trade. Major users of the OpenBazaar include buyers, sellers, and notaries. This document describes the implementation details of all major functions that create the OpenBazaar, from every type of user's perspective: buyers, sellers, and notaries.

Anticipated and Unlikely Changes

This section is intended for all possible changes that may occur to the system. They will be listed in order from most likely to least likely.

0.3 Anticipated Changes

AC1 The hardware and operating system the OpenBazaar runs on

AC2

AC3 A user's information such as their: public and private key, role (buyer, seller, notary), IP Address, Bitcoin information, digital signature, GUID, market(items,price,description), and personal settings (i.e. display picture)

AC4 The algorithm to search for nodes on the network

AC5 Personalization options for a user market

AC6

AC7

AC8

AC9

AC10

AC11

AC12

0.4 Unlikely Changes

The following are aspects of the design that are unlikely to change.

UC1 Bitcoin as a medium of exchange

UC2 Ricardian contract structure

Module Hierarchy

This section outlines the modules used in the implementation of the application. Each module is organized and decomposed according to the type of secret it contains. The following modules are represented by leaves in the hierarchy tree.

M1 GUI Module

M2 Server/Network Module

M3 Connector Module

M4 Published Contract Module

M5 Algorithms Module

M6 Active Contract Module

M7 DHT/Routing Table Module

M8 Settings Module

M9 Store Module

M10 Notary Module

M11 Bitcoin Module

M12 Initialization Module

Level 1	**Level 2**	**Level 3**
Hardware-Hiding Module	GUI Module	
	Node Module	Published Contract Module
		DHT/Routing Table Module
Behaviour-Hiding Module	Identity Module	Settings Module
		Store Module
		Notary Module
		Active Contract Module
	Algorithms Module	
Software Decision Module	Initialization Module	

Table 2: Module Hierarchy

Connection Between Requirements and Design

Module Decomposition

Below is a decomposition of each module in the application design, with details of the module's provided services and encapsulated secrets.

0.5 Hardware Hiding Modules

1. GUI Module

   - **Secret:** The underlying machine hardware and operating system environment for the application.
   - **Services:** The GUI module is responsible for handling user interaction with the system. Provides controllers which take inputted data and relay to the frontend-to-backend connector for further analysis and use.
   - **Implemented by:** The module has been partly implemented via the PyQt4 framework. Implementation will be done by creating components which inherit from classes in the PyQt4 module.

0.6 Behaviour Hiding Modules

1. Identity/Backend Module
   - **Secret:** The underlying data and behaviour requirements of the system.
   - **Services:** The backend module is primarily responsible for holding all of the modules relevant to system requirements. It holds user data including all given personalization data, trade contracts and application settings. User interaction will pass through the connector module to this module.
2. Node module
   - **Secret:** Information related to the peer-to-peer networking component of the application.
   - **Services:** This module provides all data and behaviours that make a machine a valid network node.
3. DHT Module
   - **Secret:** The contents and implementation of the distributed hash table and routing talbes for the Kademlia peer-to-peer network.
   - **Services:** The DHT module provides information about the distributed hash table used for networking. The module does node lookups and returns information about

Description

The OpenBazaar modules are broken up into logical components, abstracting away portions of the application that do not depend on one another. The first logical decomposition of the application is to abstract the details of the graphical user interface from the details of the data implementation. Each of these respective components will run as its own thread in the application environment. The data implementation can then be manipulated and accessed by the user via interaction with the GUI. To facilitate this interaction a connector module will be created. This module will expose an interface for the GUI to interact with that submits and returns data for graphical display to the user.

Front-End Client

- BazaarMain
  - Inherits from QMainWindow in PyQt4 module
  - All other GUI components are contained within the QMainWindow
  - Instance variable menuBar holds the menu bar of the application
-

Req.	Modules
R1	M1, M7
R2	M1, M7
R3	M6, M9
R4	M3, M9
R5	M3, M9
R6	M4, M6
R7	M4, M6
R8	M10
R9	
R10	
R11	
R12	M10
R13	M10

Table 3: Trace Between Requirements and Modules

Back-End Server

-

Server-Client Connection

-

Traceability Matrix

Below are two traceability matrices. The first demonstrates the connection between the functional requirements and modules while the second describes the connection between the anticipated changes and modules.

Use Hierarchy Between Modules

Detailed Timeline

Gantt Chart

Pert Chart

References

Modules for GUI

AC	**Modules**
AC1	M3
AC2	M3
AC3	M8
AC4	
AC5	
AC6	
AC7	
AC8	
AC9	
AC10	
AC11	
AC12	

Table 4: Trace Between Anticipated Changes and Modules

QMainWindow - RedevBazaar

Holds everything else in the application

RedevBazaarConnector

QVBoxLayout

Main layout for two windows

QLabel - Logo

Holds main logo, top right corner

Avatar photo - QLabel

Holds any chosen avatar logo from the user

QLineEdit - Search bar

QTabWidget

Holds the menus for messages, etc

QButton - Search

QListWidget - Merchant list

QLineEdit - Add merchants

QButton - Add Merchant confirm

QListWidget - Notary list

QLineEdit - Add Notary

QButton - Add notary confirm

QListWidget - Past contracts

Modules for Server Networking

The server end of the application has a networking component to enable the decentralization of the application.

This component includes:

Unique GUID for network node

Create a public key between 1 - $2^256, deriveprivatekeyfromthat$

Sign own public key with private key

SHA256 this self-signed key

RIPEMD160 the SHA256 hash

Distributed hash table

Row would be GUID, dynamic IP, port, contract hash, keyword

Contains list of nodes within similar range. Allows for easy finding of other network nodes

Ricardian Contracts Metadata

Info about contract creation date, valid period etc

ID Module

Contains all user info as given by the user

Trade module

Contains information regarding services/goods provided and bitcoin amount received

Ledger Module

Contains info created as the contract is signed and processed, to ensure validity