Contents

[1. Description of the project 2](#_Toc436512384)

[1.1 Goal 2](#_Toc436512385)

[1.2 Product 2](#_Toc436512386)

[1.3 Process 4](#_Toc436512387)

[2. Product History 5](#_Toc436512388)

[2.1 Release 5](#_Toc436512389)

[2.2 Adoption 5](#_Toc436512390)

[2.3 Feature Tracking 5](#_Toc436512391)

[2.4 Critical Reviews 5](#_Toc436512392)

[3. Product Structure 6](#_Toc436512393)

[3.1 Architecture 6](#_Toc436512394)

[3.1.1 Layered Architecture 6](#_Toc436512395)

[3.1.2 Key Modules of core framework 7](#_Toc436512396)

[3.2 Design 8](#_Toc436512397)

[3.1.1 High Level Design 8](#_Toc436512398)

[3.1.2 Detailed Design 8](#_Toc436512399)

[3.1.3 Domain Model Design 8](#_Toc436512400)

[3.1.4 DB Design 8](#_Toc436512401)

[3.3 Code-base 9](#_Toc436512402)

[3.4 Documentation 10](#_Toc436512403)

[3.5 Test 10](#_Toc436512404)

[3.1.5 Unit Testing 10](#_Toc436512405)

[3.1.6 Integration Testing 12](#_Toc436512406)

[4. Defect Management 13](#_Toc436512407)

[4.1 Report 13](#_Toc436512408)

[3.1.7 Resources 13](#_Toc436512409)

[3.1.8 Required information to report an issue 13](#_Toc436512410)

[4.2 Repair and Release 13](#_Toc436512411)

[5. Appraisal of the Product and Management 15](#_Toc436512412)

[5.1 Standards Compliance 15](#_Toc436512413)

[5.2 Maintainability 15](#_Toc436512414)

[3.1.9 Find Metrics 15](#_Toc436512415)

[3.1.10 Readability 15](#_Toc436512416)

[3.1.11 Analyzability 15](#_Toc436512417)

[5.3 Reliability 15](#_Toc436512418)

[5.4 Usability 15](#_Toc436512419)

[5.5 Extensibility 15](#_Toc436512420)

[5.6 Portability 15](#_Toc436512421)

[5.7 Scalability 15](#_Toc436512422)

# Description of the project

## Goal

BroadleafCommerce is an open-source, e-commerce framework written entirely in Java on top of the Spring framework. It is targeted at facilitating the development of enterprise-class, commerce-driven sites by providing a robust data model, services and specialized tooling that take care of most of the "heavy lifting" work. To accomplish this goal, we have developed our platform based on the key feature sets required by world-class online retailers - and we're committed to continually expanding our feature offering. We've also taken extra steps to guarantee interoperability with today's enterprise by utilizing standards wherever possible and incorporating best-of-breed, open-source software libraries from the community.

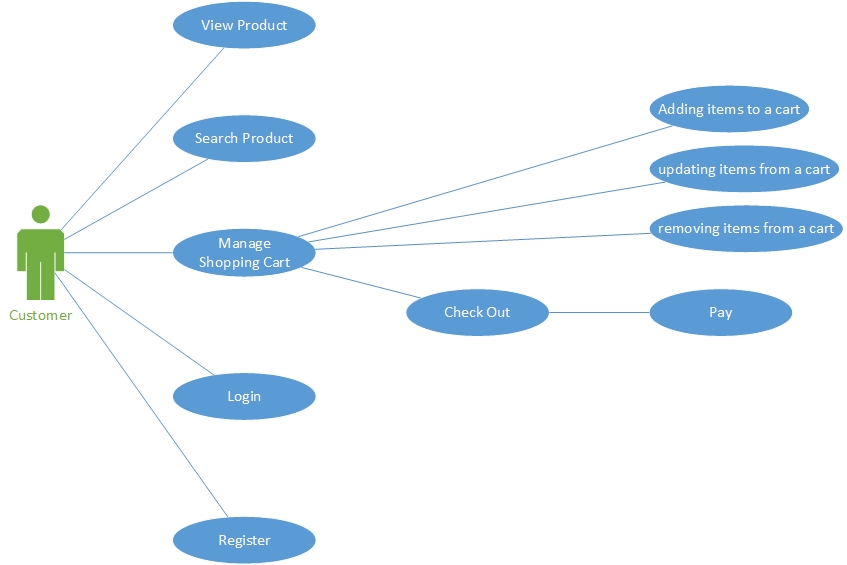
## Product

Product List:

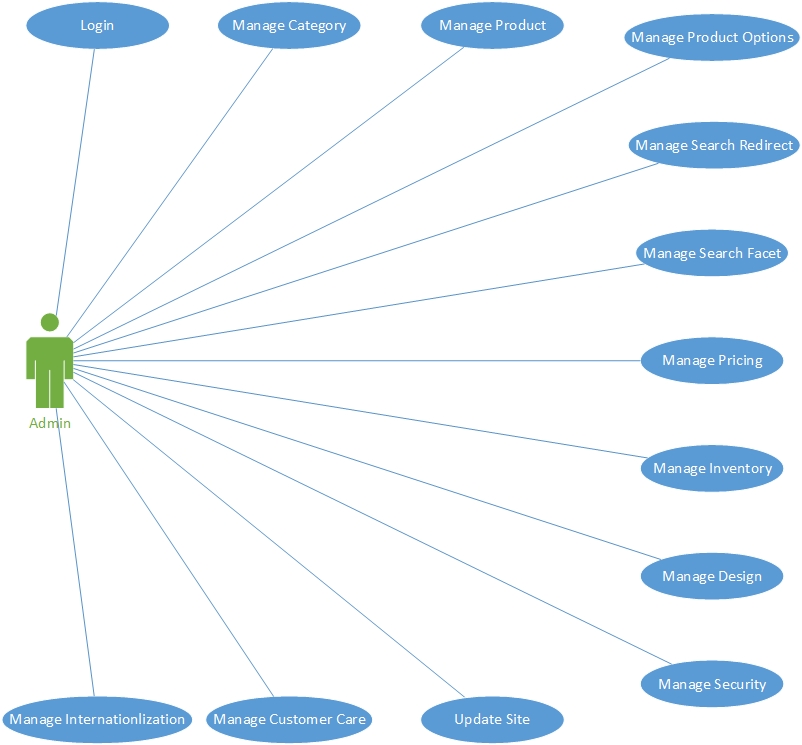
|  |  |
| --- | --- |
| Item | Description |
| blc-authorizenet | Broadleaf Authorize.net Integration Module (Direct Post Method) |
| docs | The new Broadleaf wiki |
| Menu | Broadleaf Commerce currently offers a community abstraction Menu module that allows managing Categories and Site Navigation in terms of Menus in the Admin much easier. |
| Demo Site | The New Broadleaf Commerce Demo Site |
| BroadleafCommerce | Main Project |
| [blc-amazon](https://github.com/BroadleafCommerce/blc-amazon) | Integration of BLC and Amazon services. |
| [copy-github-labels](https://github.com/BroadleafCommerce/copy-github-labels) | Easily copy GitHub labels from one repository to another. Uses [GitHub API for Node.js](https://github.com/mikedeboer/node-github). |
| [blc-paypal](https://github.com/BroadleafCommerce/blc-paypal) | Broadleaf PayPal Integration Module (Express Checkout) |
| [ansible](https://github.com/BroadleafCommerce/ansible) | Ansible is a radically simple IT orchestration engine that makes your applications and systems easier to deploy. Avoid writing scripts or custom code to deploy and update your applications— automate in a language that approaches plain English, using SSH, with no agents to install on remote systems. |
| [ModuleTemplate](https://github.com/BroadleafCommerce/ModuleTemplate) | A template project for creating Broadleaf modules |
| [BasicInventory](https://github.com/BroadleafCommerce/BasicInventory) | Basic inventory module |
| [CatalogMetaData](https://github.com/BroadleafCommerce/CatalogMetaData) | Broadleaf MetaData Module. |
| [blc-usps](https://github.com/BroadleafCommerce/blc-usps) | Broadleaf USPS Integration Module |
| [blc-cybersource](https://github.com/BroadleafCommerce/blc-cybersource) | Broadleaf CyberSource Integration Module |
| [I18n](https://github.com/BroadleafCommerce/I18n) | Internationalization Extension Module |
| [AdminLanguagePack](https://github.com/BroadleafCommerce/AdminLanguagePack) | Project to hold Admin locale specific property files |
| [MavenArchetypes](https://github.com/BroadleafCommerce/MavenArchetypes) | Maven archetypes for jump starting a Broadleaf Commerce implementation |
| [GrailsPlugin](https://github.com/BroadleafCommerce/GrailsPlugin) | broadleaf-grails-plugin |

Main Project:

Use-case Diagram Customer:



Use-case Diagram Admin:



<http://www.broadleafcommerce.com/features#cartAndCheckout>

<https://www.youtube.com/watch?v=1xVF8xIrbcI>

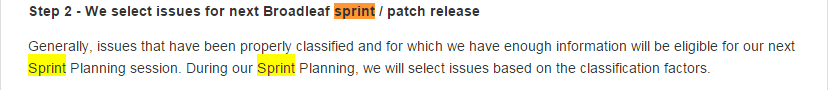
<https://www.youtube.com/watch?v=EiuY3bzzVbw&feature=youtu.be>

## Process

Agile Scrum:



Evidence:



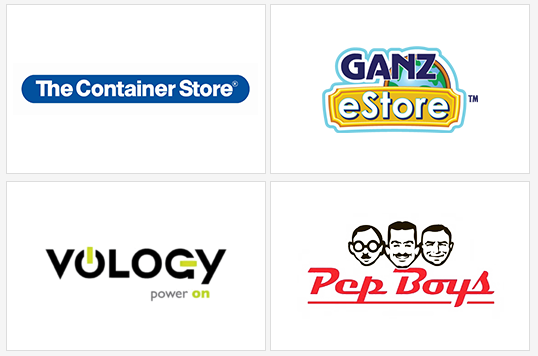
Hyperlink: https://github.com/BroadleafCommerce/BroadleafCommerce/blob/BroadleafCommerce-4.0.x/CONTRIBUTING.md

# Product History

## Release

## Adoption

BroadleafCommerce is proud to power the eCommerce capabilities for some of the most recognized brands around the globe. One thing all of their clients share, is the need to have a commerce that works for their unique commerce needs. No two businesses are exactly alike, so this system is provided to celebrate that fact. Following are some clients running their business with system built on BroadleafCommerce framework.



* The Container Store

Home/Office supplies, Top 500 Internet Retailer Site

* Ganz eStore

Broadleaf Commerce allowed Ganz to launch a re-vamped, elegant, effective eCommerce solution that meets all of their complex business and technical requirements.

* Vology

Network Hardware store, B2B provider selling approximately $140 million of technology solutions each year to small and mid-sized companies, Fortune 2000 businesses, government agencies, as well as educational institutions and districts nationwide.

* Pep Boys

Auto parts store with 99,417 Unique SKUs, 723 Stores and 30+ Back-end Integrations

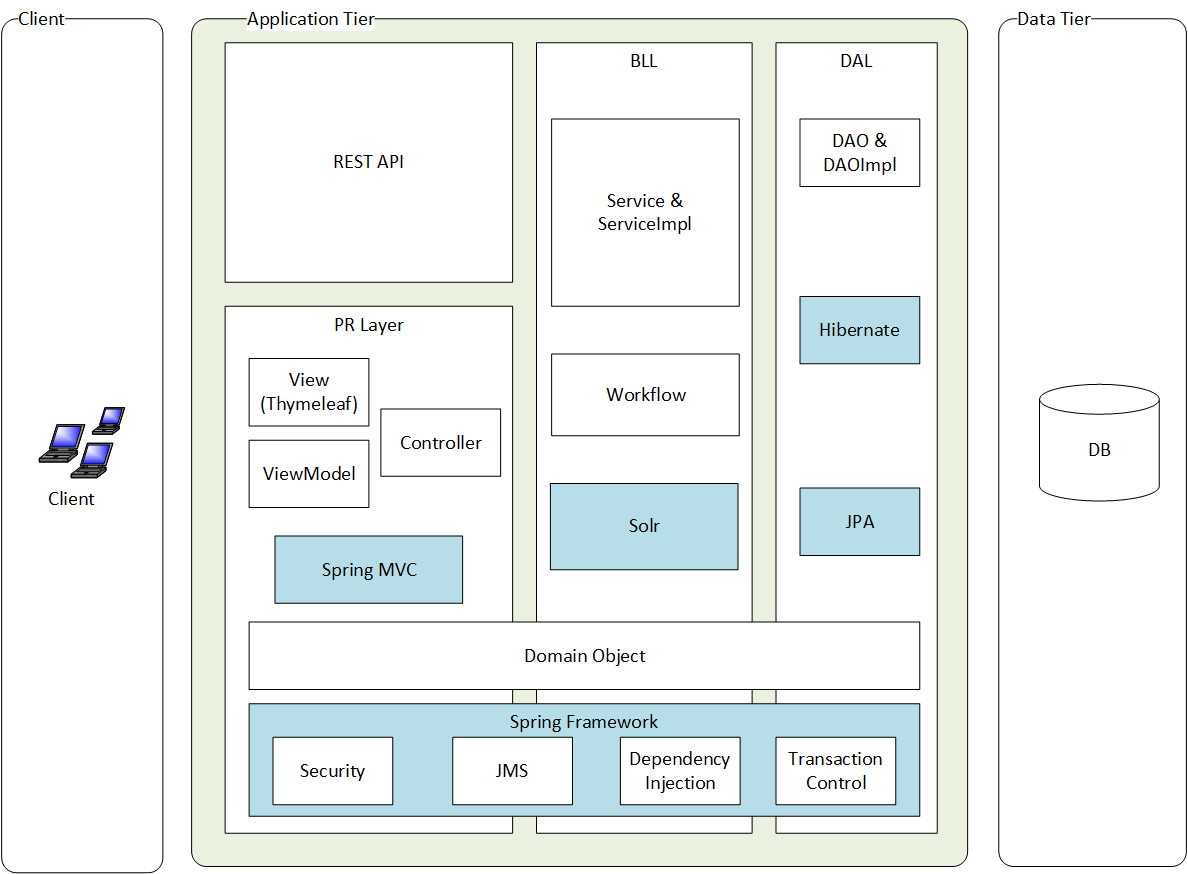
## Feature Tracking

## Critical Reviews

# Product Structure

## Architecture

### Layered Architecture



Broadleaf follows Sun J2EE guideline, its application comprises Presentation Layer, Business Logic Layer and Data Access Layer. Spring Framework provides the component architecture used by Broadleaf Commerce. All three layers has dependency on Broadleaf Domain Models.

#### Presentation Layer

The web view layer of out of box Broadleaf Commerce application is built on Spring MVC framework. The Thymeleaf template engine which is well-integrated with Spring MVC is recommended for processing and generating HTML, XML, JavaScript, CSS and text instead of JSP.

Besides web view, Broadleaf Commerce exposes a set of functional APIs as RESTful services that allow integration with other applications, including mobile applications. These APIs are easy to expose, easy to extend, and provide a simple pattern for adding net new functionality. XML and JSON formats for messages are supported.

#### Business Logic Layer

Besides regular Online Commerce Domain Business Logic implemented as Service and ServiceImpl, there are some other features provided in this layer

* **Configurable Workflow**, as a Key areas in the eCommerce lifecycle. Implementers have full control over the keys steps in pricing and checkout, allowing manipulation of module order, behavior, and custom execution. Composite workflows are also supported to achieve sophisticated, nested behavior.
* **Task Scheduling**, Repetitive tasks can be scheduled through the Quartz job scheduling system.
* **Search**, Flexible domain search capabilities are provided through integration with Solr, a popular Compass and Lucene projects.

#### Data Access Layer

Broadleaf Commerce implements Java Persistence API using Hibernate Object-relational mapping Framework.

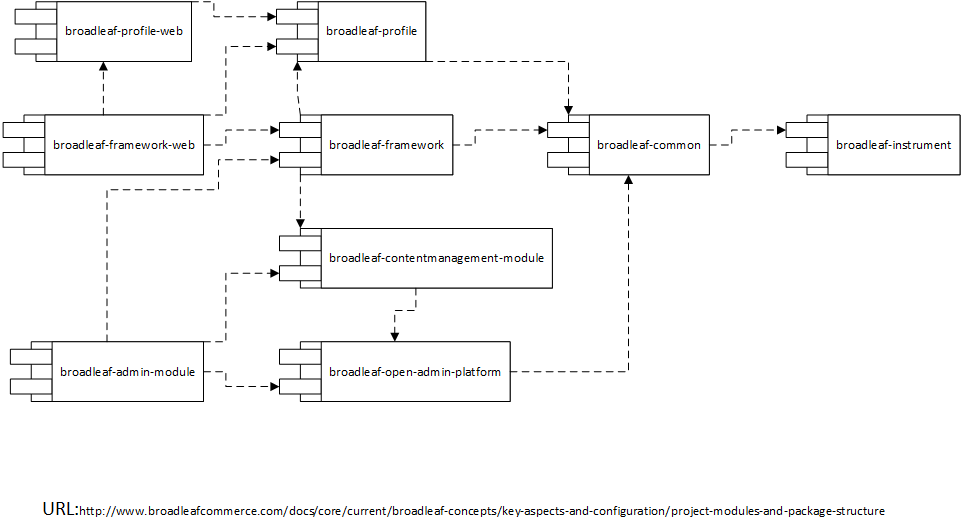
#### Crosscuttings

All above layers have dependency on Broadleaf Commerce Domain Model to transfer data or perform operations.

Broadleaf Commerce takes advantages of numerous features Spring Framework provided, such as

* Spring Security for controlling authentication and authorization at both the code and page level
* Spring JMS for asynchronous processing of application messages
* Dependency Injection and Inversion of Control
* AOP and transaction control

### Key Modules of core framework



There are currently 9 key modules included as part of the Broadleaf Commerce framework. It would be exceptionally rare for a Broadleaf Commerce implementation NOT to use all of these core modules. The about diagram also shows their dependency relationship.

#### Framework

This generically named module represents the commerce functionality of Broadleaf Commerce (e.g. Orders, Products, Offers, etc.

* broadleaf-framework, Core Broadleaf framework classes
* broadleaf-framework-web, Spring MVC controllers and related items

#### Profile

This module provides the concept of Customer. It is provided separately from framework in anticipation that some may want to utilize these features without using the Commerce features.

* broadleaf-profile, Customer profile related classes, utility classes, email, configuration merge
* broadleaf-profile-web, Spring MVC controllers and related items supporting the profile module

#### CMS

This module provides content management functionality that supports targeting ad based content to users based on their profile as well as static page management.

* broadleaf-contentmanagement-module, A full-featured content management system that is managed via the administration tool

#### Open Admin

The Broadleaf Commerce admin architecture which allows annotated JPA entities to be administered via a rich UI. The intent of separating this module is to provide some architecture purity while leaving open the possibility that the admin techniques used by Broadleaf Commerce may be provided outside of the Commerce application in the future.

* broadleaf-admin-module, Contents: Broadleaf Commerce specific administration module that plugs into the open admin platform
* broadleaf-open-admin-platform, Framework for creating extensible administration GUIs for Hibernate managed domains

#### Common

* broadleaf-common, A collection of classes shared by various modules.
* broadleaf-instrument, Allows for runtime instrumentation to override certain Broadleaf annotations

## Design

### High Level Design

As an out of box product as well as a framework for enterprise eCommerce, high level design document is provided with

* Logical view, help user to identify key components and their responsibilities
* Physical view, help user to understand the possible approach to deploy the software on physical devices and network environment.
* Key architectural decisions such as
  + Choosing the Spring Framework over EJB3
  + Using Solr to achieve catalog browsing and searching

### Detailed Design

From the documentation provided, there is no evidence that they followed the RUP Object-Oriented Analysis and Design Approach and did detailed level design before wrote the code. Our assumption is that they follow the Agile manifesto: **Working software** over comprehensive **documentation.**

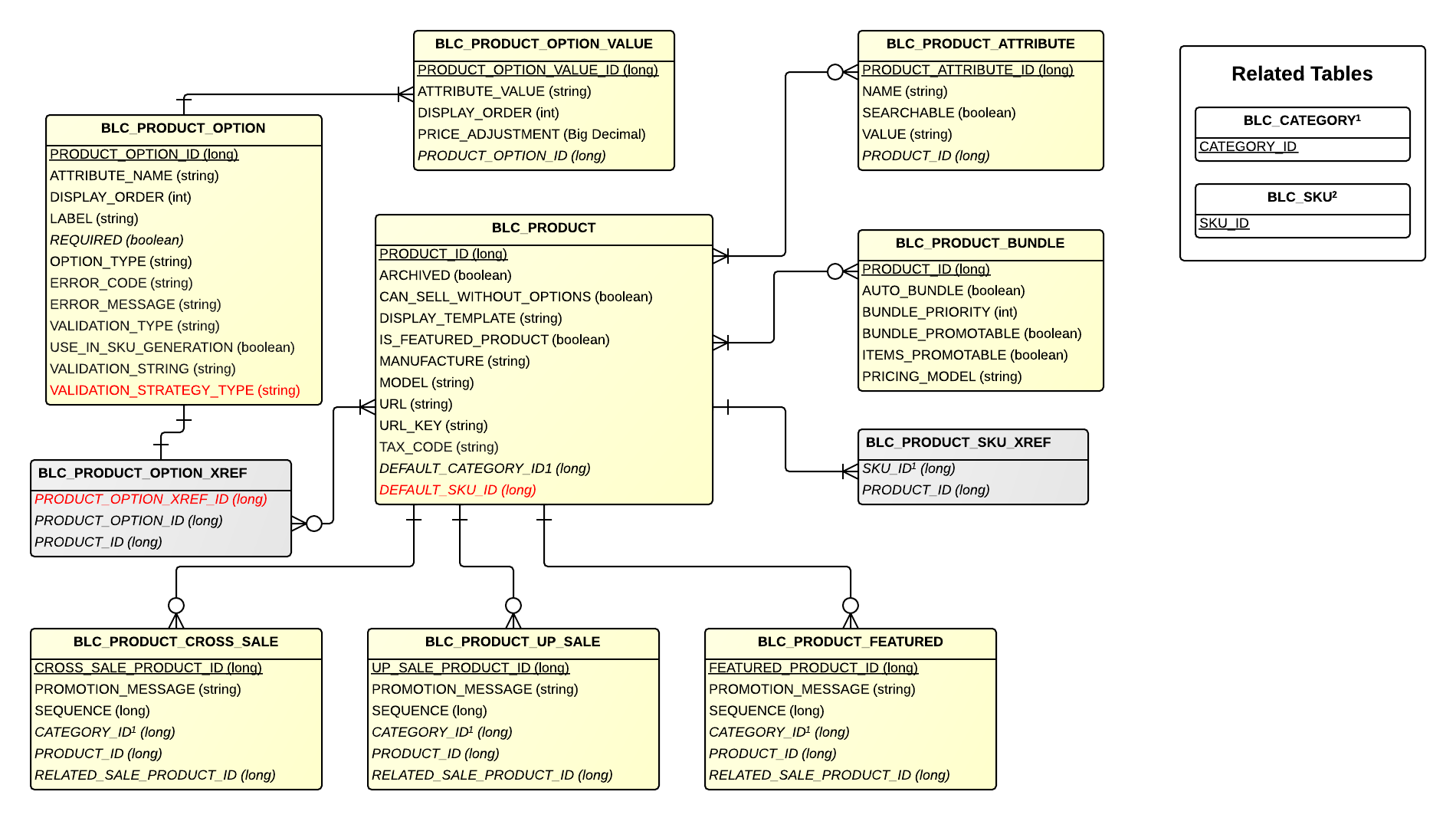
### Domain Model Design

In the source code we discover quite a lot of domain classes found for each feature such as catalog, checkout, inventory and offer etc. Under org.broadleafcommerce.core.catalog.domain package, there are 48 interfaces and classes, including Category, CategoryAttribute, Product, ProductAttribute etc.

However, there is no Domain Model Diagram provided which means that to fully understand it, we have to do reverse engineering to get the class diagram from codes. And with that large amount, it is quite troublesome.

### DB Design

In terms of DB Design, perhaps in order to compensate the lack of Domain Class Design, ER diagrams are satisfactorily comprehensive and in detail. As an example, ER diagram of Catalog Product.



## Code-base

Although we are not able to find the design guideline, detailed design or class diagram, our observation is

* Packages and classes are Well-organized.

For instance, in BroadleafCommerce Framework modules, basically for each feature like catalog, there are dao, domain, service, service.exception etc. which is very helpful for user to expect what class can be found in that package.

* SOLID principles are followed

In the Product entity class, it has an aggregation of objects of ProductAttribute as attribute. Hence every attribute of a product is defined outside the Product class which makes it very flexible to change, add or remove from a product. It illustrate the Single Responsibility Principle and Open-Close Principle.

Furthermore, the ProductAttribute is an interface, its implementation class ProductAttributeImpl has been hiding from Product. It shows the Dependency Inversion Principle.



## Documentation

Following documentations are provided:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Getting Started | Getting environment up and running, show where things live, and walk through a few examples. |
| Broadleaf Concepts | This section is where the majority of information on Broadleaf functionality and concepts can be found. We describe how important operations such as cart modification, pricing, and payment occur as well as additional configuration that Broadleaf supports. |
| Modules | Broadleaf Commerce can be enhanced with add-on modules. |
| Tutorials | This tutorial section will walk you through examples start to finish that demonstrate how to work with various different parts of Broadleaf Commerce. |
| Database Model | Model Changes and current ER Diagram |
| Release Notes | The note of release, including description, bugs, and enhancements. |
| Migration Notes | This section contains migration documents that will help developer transition his current Broadleaf installation to a more recent version. |
| Appendix | Other documents which are associated with the project. |
| Javadocs | Javadocs including Package, Class, Use, Tree, Deprecated, Index, and Help. |

BroadleafCommerce comes with quite a lot of documentations to support users, most cases are developers, although it is not so clearly structured. It is not easy for a new comer to find out what he wants.

For example Getting Started and Tutorials are similar from the name.

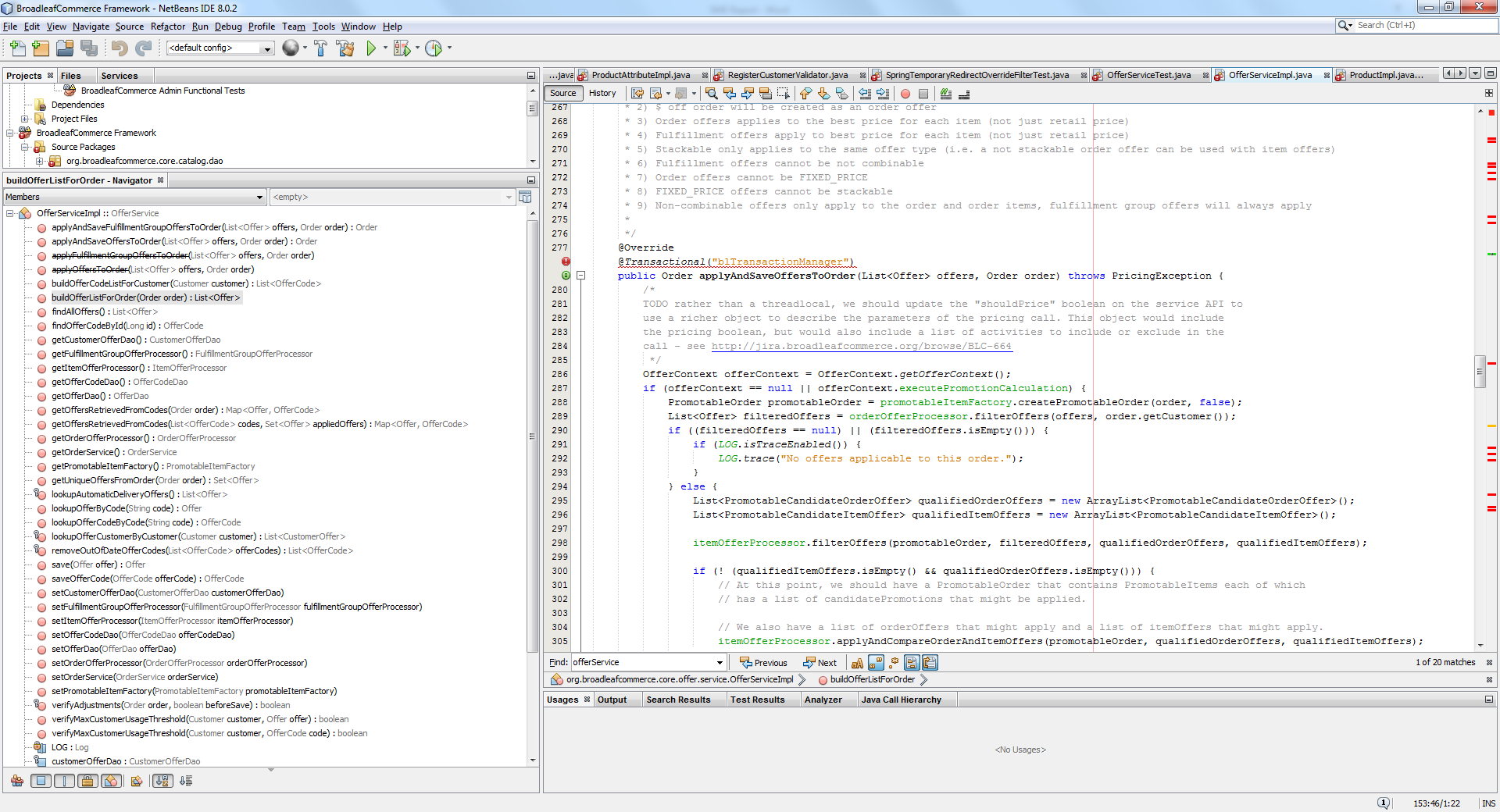
Moreover, under “Broadleaf Concepts” menu item, there are many items like “Merge Configuration”, “Persistence Configuration” and “Database Configuration” within “Key Aspects & Configuration” catalog. Besides that there is an “Additional Configuration” in the same level with “Key Aspects & Configuration”.

## Test

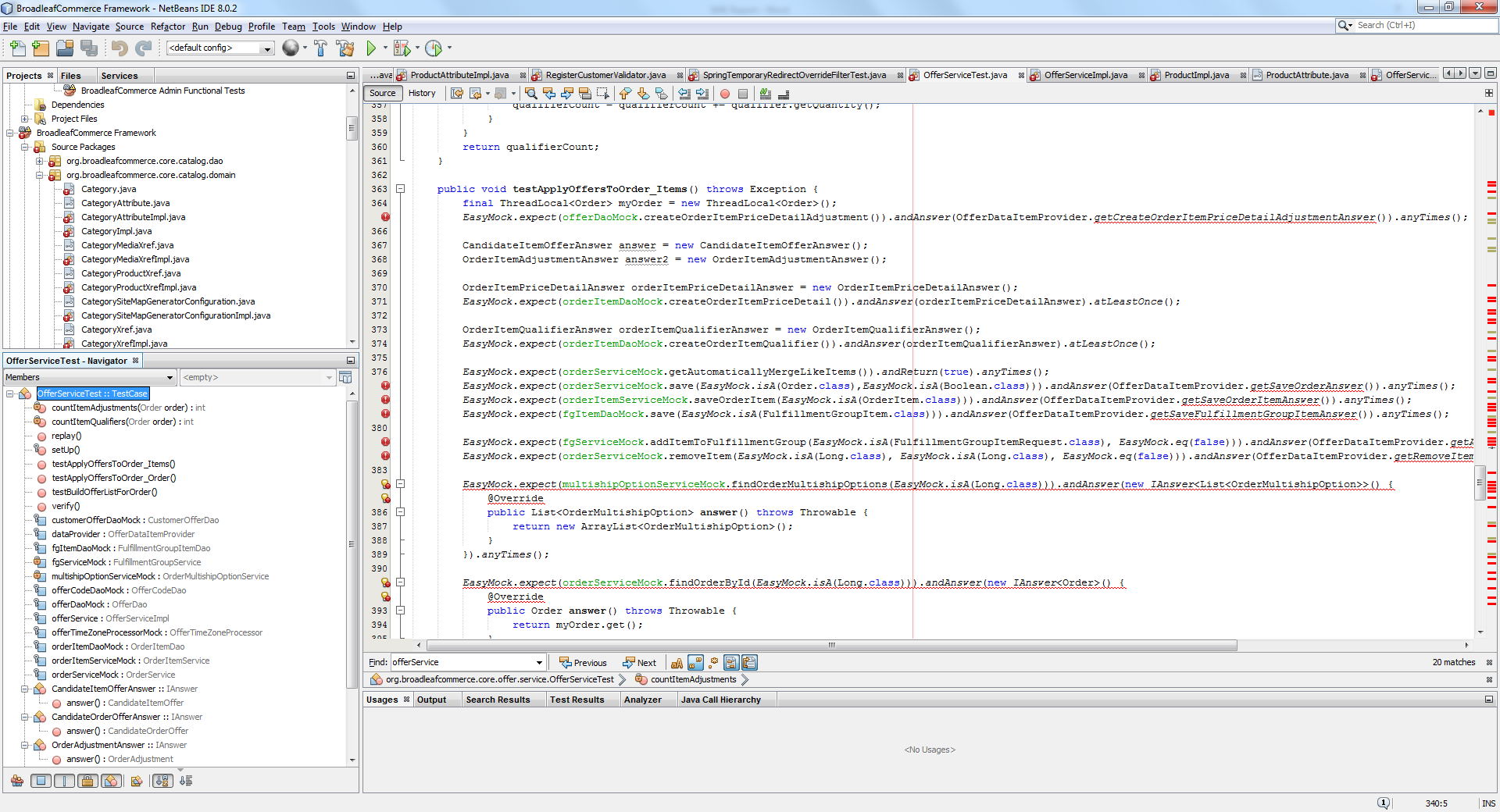
### Unit Testing

As a J2EE application, BroadleafCommerce uses JUnit and a mocking framework called EasyMock to perform unit test.

According to published source code, the Unit Test strategy is only testing significant method of business logic classes. For instance, the diagram below shows the methods of OfferServiceImpl class.



Only two of all methods have been tested.



Due to the widely applied Dependency Injection, some dependencies created, managed and injected into the object by context. To make the class testable, setter method of those dependencies are exposed as Object Seams to enable manipulation of their behaviors by setting mockups into the object.

@Service("blOfferService")

public class OfferServiceImpl implements OfferService {

@Resource(name="blCustomerOfferDao")

protected CustomerOfferDao customerOfferDao;

@Resource(name = "blOrderService")

protected OrderService orderService;

…

@Override

public void setCustomerOfferDao(CustomerOfferDao customerOfferDao) {

this.customerOfferDao = customerOfferDao;

}

@Override

public void setOrderService(OrderService orderService) {

this.orderService = orderService;

}

…

}

### Integration Testing

Spock/Groovy is introduced to perform Integration Tests. And with the 4.1 release of Spring, developers now have the ability to test RESTful services, workflows and other integration test abilities in a much simpler format.

To test a RESTful services, first step is to set up the test spec by with POM and plugin. Then it requires Spring-Test's MockMVC api with its annotations as well as Groovy language. But without the expertise of those technologies, we found that is quite difficult to understand.

# Defect Management

## Report

### Resources

There are two approaches provided for BroadleafCommerce community users to report defects.

* Forum (<http://forum.broadleafcommerce.com/>)

Forums are useful for requesting help from other community members.

* GitHub Issue tracker

GitHub issues are used to track bugs.

In another words, when a user is uncertain whether BroadleafCommerce provides a feature he expects or if he encounters is the intended behavior or a bug, he can go to the forum. Otherwise he should raise an issue on GitHub Issue tracker.

### Required information to report an issue

When user wants to raise an issue, following information is required to help the team reproduce and locate the defect

* Broadleaf version you are using
* Steps to reproduce
* Any stack traces that you receive (if applicable)
* Any additional information that allows us to help you faster
* Any potential fixes you might have already tested

## Repair and Release

There is a well-defined process for defect repair and release in CONTRIBUTING.md file.

* Step1 – Analyze the issue

Team member will first analyze the issue and label the raised issue.



Here are the category of labels

* + Severity-(critical/major/minor)
  + Module-(admin/cms/core/rest/tests)
  + Type-(bug/enhancement/feature).
  + Difficulty-(hard/medium/easy)
  + Affects
  + Target to Release
* Step2 - select issues for next Broadleaf sprint / patch release.

During the sprint planning, team member will select the issues to be fixed and release according to their priorities.

* Step3 - assign a milestone indicating the release that targets for the issue



* Step4 - mark the issue as closed generally once the developer has committed the fix
* Step5 - We release the patch (or new major version) of the software

In some cases, due to some resource or schedule constraints, the issues reported cannot be repaired within a certain time period, which may cause inconvenience to users. Hence an alternative way recommended by BroadleafCommerce team is that since it is an open-source product user can fix the defect then send a pull request in GitHub after passing unit test and regression test. The team will review the validity and merge the changes to be released.

That is also the reason why open source software hardly shows declining quality over time as what we have learned from software maintenance and evolution course. With the large amount of users who are keen to contribute to improve the software, defects are identify and repair effectively.

# Appraisal of the Product and Management

## Standards Compliance

## Maintainability

### Find Metrics

• Abstractness

• Cyclometric complexity

• Weighted methods

• Find ant design pattern usage

### Readability

Check readability and comment on the findings

### Analyzability

Server Log,QoS is not Traceability

## Reliability

## Usability

## Extensibility

Look into Extensibility

Propose strategy to support the Extensibility

<http://www.broadleafcommerce.com/docs/core/current/broadleaf-concepts/key-aspects-and-configuration/features-and-architecture>

extend entity, service

## Portability

3. Check the Hibernate does it support query optimization with other DB

## Scalability