Design Document Final Version

phpGroupWare Web Content Manager

ICS 125
Summer 2002 – 10-Week Session
Professor Hadar Ziv

Team 10 – Project X
http://tina.alinaghian.com/ics125/index.html
Tina Alinaghian (57291129)
Patrick Walsh (91105649)
Fang Ming Lo (52013421)
Austin Lee (50792270)
Siu Leung (51390400)

TABLE OF CONTENTS

SECTION 1: OVERVIEW

Project Introduction

Team 10 will build a "Collaborative Web Content Management" application based on the phpGroupWare platform. When complete, the application will be installed on the phpGroupWare web site to demonstrate its capabilities and test it in a real-world environment.

The system will allow a community of individuals to contribute to a web site by granting individual members or groups within the community permission to view and edit specific sections of the web site. In general, the viewers of the web site function in one of three roles: Web Site Administrators, Web Site Contributors, and Web Site Viewers (may be anonymous). For short, we will refer to these roles as Administrators, Contributors, and Viewers. Administrators are the all-powerful beings that manage the entire site. Contributors are authenticated users with permissions to manage specific sections of the web site. Viewers are authenticated or unauthenticated viewers of the site. Sections of the web site, also called Categories, have permissions that determine whether anonymous viewers may see them, and which authenticated users or groups may view or edit them. Administrators may always view and edit all categories on the web site.

The Administrator's functions are:

1. Manage categories and view/edit permissions.
2. Manage the overall look and feel of the site including constant site-wide content.

The Contributor's functions are:

1. Addition/ deletion of pages in an existing category.
2. Modification of the content in existing pages.

The Viewer's functions are:

1. Browse the generated web site.

The entire web site will be dynamically generated from a database. Hence, there will be a Page Generation Engine that generates web pages based on the Administrators' specified look and feel and the Contributors' content. The Page

Generation Engine will evaluate whether the Viewer has permissions to view a particular page before generating it.

The entire application will be written using the phpGroupWare backend. This means that the list of users, groups, and administrators will be taken from an existing phpGroupWare installation. phpGroupWare will handle the management of these users and groups. phpGroupWare will also handle the authentication of users and track them through sessions. phpGroupWare will also be responsible for storing and retrieving all data that needs to be saved. This includes all of the content, the list of categories and permissions, etc. through phpGroupWare's database abstraction functions.

Finally, the completed application will have some compatibility with an already established web application of similar functionality. PHPNuke is a program used to generate dynamic web sites. It was built to allow third parties to create add-ons called "blocks" that can be added to a web page. These PHPNuke blocks are small pockets of information that usually sit on the side of a web page. For example, a block may contain a list of stocks and their current prices in a box on the left side of a page. In addition, PHPNuke supports themes, or pre-fabricated web site designs. There are a wide variety of these themes and blocks already available. This application will support both PHPNuke themes and PHPNuke blocks.

SECTION 1: OVERVIEW (CONT.)

Design Overview

The phpGroupWare Web Content Manager project is divided into two sub-applications, with some common functionality shared between them. There is first the public facing application that generates the web site that the average viewer sees. This "sub-application" or package is called "wcm-pub." Wcm-pub uses the phpGroupWare API, but does not run within the phpGroupWare environment, as do normal phpGroupWare applications. It will reside in a directory outside of the phpGroupWare directory structure and will have access to only a small fraction of phpGroupWare's overall functionality.

The second "sub-application" (or package, if you prefer) is a true phpGroupWare application, running fully inside the phpGroupWare environment. Only non-anonymous users with valid phpGroupWare accounts and specific sets of permissions can use this application. Administrators and Contributors who change the content or look and feel of the site will use this application. This "sub-application" is called "wcm-gw."

Finally, because the two sub-applications share a certain amount of functionality, there needs to be a set of common classes that both applications draw on. These have been lumped into a package called "wcm-common."

Every class is associated with one of these three packages. Each class is also associated with a single "tier" of a three-tiered architectural model. In Microsoft's language, this is called, "Model, View, Controller." In Rational's language, this is called, "Entity, Boundary, Control." In phpGroupWare applications, the same concept is also used (and, in fact, enforced), but under yet a different name. Classes that manage data in phpGroupWare are called *Storage Objects*, or SO for short. These classes correspond to Model or Entity classes in other jargons. Classes that manage the logic and control flow of an application are called *Business Objects*, or BO for short. These classes correspond to Controller or Control classes in other jargons. Finally, classes that manage interactions with the end-user – displaying data and accepting input – are called

User Interface objects, or UI for short. These classes correspond to the View or Boundary objects of other jargons.

Nearly every Storage Object class will make use of the phpGroupWare database interface classes for storing and retrieving database data. Nowhere else should this happen.

Nearly every Business Object class will make use of a single ACL.BO class, which will make use of phpGroupWare's user, group, and access control list classes.

Nearly every User Interface class will make use of either phpGroupWare's template system or our own phpNuke theme compatibility system for managing the look and feel, positioning, and overall display of user interface elements and for completely separating HTML from even the UI classes. This way, new HTML "skins" can be swapped in and out without touching a single class.

SECTION 2: PROJECT PLAN (CONT.)

Project Risks and Challenges

A. Knowledge

i. 3 of the 5 team members have either limited experience or no experience at all developing web applications and particularly programming in PHP. This poses a large risk as the learning curve associated in getting up-to-speed enough on PHP and web application development is hard to predict. As such, the time required for team members to complete certain coding tasks will be highly variable. With the short time frame of the project, this presents a monstrous risk.

ii. 4 of the 5 team-members have no experience using phpGroupWare. As the project is starting now, it is obvious that this is one of the biggest challenges. ***Update:*** Team members are having difficulty learning and using the phpGW API.

iii. In addition to the basic programming language, 4 of the 5-team members have not had experience using collaborative development tools such as the Concurrent Versioning System (CVS) or Microsoft Visual Source Safe. This will potentially slow development down and create a difficulty in collaborating between team members. ***Update:*** So far this is not proving to be a big problem.

iv. 4 of the 5 team members have limited or no experience using Linux machines. Since the web server and development will be taking place on a Linux machine, team members will also have to pick up at least a rudimentary understanding of Linux in order to develop for the project. This again may take up more time than predicted. ***Update:*** So far this is not proving to be a big problem.

v. 3 of the 5 team members have limited or no experience with SQL and relational databases. Again, this may have a negative effect on the team's ability to get the application finished on time.

B. Customer Constraints

i. The customer, Dan Kuykendall, the leader of the phpGroupWare project and our technical adviser, is currently starting a new job, raising two children including a three-week-old baby, and squeezing us in when he can. This means that he is not as available to us as we would like and, as the project moves

forward and we need help and advice on technical matters, this may become a problem. ***Update:*** Dan has been extremely difficult to get a hold of, as feared.

C. Personnel Constraints

i. All of the team members are either working part-time or full-time jobs or attending other summer school classes in addition to ICS 125. This means that the time that each team member has to dedicate to the project is finite and potentially more limited than is required to adequately finish the project in the allotted time.

D. Beta Platform

i. phpGroupWare is the platform of choice for this project. It serves almost as an operating system, handling the low level details such as data storage and retrieval and user authentication and permissions. However, phpGroupWare is not a released product and is currently in Beta testing. This project is being built on a Beta Release Candidate, meaning that the code is not entirely stable and far from bug-free. This may not be an issue at all, or we may run into problems with phpGroupWare that will sidetrack us from the project at hand and prevent or delay an adequate completion of the project.

E. Lack of Documentation

i. phpGroupWare and phpNuke both have limited up-to-date documentation for developers. This means that much of the time we will have to reverse engineer these programs and/ or attempt to talk with various developers, who may live half way around the world, to resolve problems or issues. ***Update:*** So far, this is proving to be the biggest challenge and the biggest slowing factor in development.

F. Time Constraints

i. All of the above risks are risks that the project might not be done in time, rather than that it might not be done at all. Given sufficient developer time and attention, this project will certainly be completed. However, this project has a very strict 10-week time line. In fact, the coding portion of the project will last at most 4 weeks – a short amount of time to code, integrate, and test five semi-independently developed portions of code.

These risks are all real and any one of them could be fatal to the project. Only disciplined management, clear and uninterrupted organization, a well-defined

design document, and careful attention to the set schedule can overcome these risks. Also, if any one-team member gets stuck somewhere, other team members will have to help in order to keep the project moving forward.

SECTION 2: PROJECT PLAN (CONT.)

Project Resources

Technical and Personal Team Resources:

Team 10 consists of five talented individuals all with their own unique personal and technical strengths. These strengths will be utilized to accomplish our goals in this project. Bellow are the personal and technical strengths of each member on our team:

Patrick Walsh

Technical Strengths: Patrick has a varied background in computers that includes programming in numerous programming languages. He has experience working with PHP, MySQL, phpGroupWare, Apache, and the other platforms being used in this project.

Personal Strengths: Patrick spent the past four years running a software company where he designed and oversaw the development of an enterprise-level software product that was successfully brought to market. This has contributed to his experience in working to collaboratively develop a software application – experience that can be drawn on directly for the WCM project.

Tina Alinaghian

Technical Strengths: Tina has proficient knowledge of web page design and development. She has experience with many database platforms including mySQL and msSQL. She also has experience with HTML, PHP, ASP, Java, and C++.

Personal Strengths: Tina is a hard worker and a very quick learner.

Siu Leung

Technical Strengths: Siu is an adequate programmer with knowledge in Java and C++. Although he is not familiar with PHP, he is excited to pick up a new programming language.

Personal Strengths: Siu is able to socialize and work well with others both inside and outside of a group project-type environment. He is an organized and responsible individual who finishes assigned tasks on time.

Austin Lee

Technical Strengths: Austin is a programmer with knowledge in Java, Visual Basic, C++, and other languages. Although he is not familiar with PHP, he can quickly pick up new programming languages.

Personal Strengths: Austin had been a researching intern in a research center in Korea (Samsung Advanced Institute of Technology). His solid mathematical background could help others with logical problems.

Fang Ming Lo

Technical Strengths: Ming is knowledgeable in Java and C++. He has also used HTML in the past, and although he may be a bit rusty with it, he is confident he can pick it up fast. He is currently trying to learn .NET and is interested in learning the new and useful languages currently out.

Personal Strengths: Ming is very organized and can work well in a group. He takes his assigned jobs very seriously. He is also a very easygoing person and very easy to get along with.

Software and Hardware Resources:

We will also be utilizing the following software on our web and development server to complete the project:

1. Mandrake Linux 8.2
2. Apache-AdvancedExtranetServer 1.3.23
3. PHP 4.1.2
4. phpGroupWare 0.9.14RC3
5. OpenSSH 3.4p1
6. CVS 1.11.1p1
7. MySQL 3.23.47

SECTION 2: PROJECT PLAN (CONT.)

Staff Organization

Team Members:

Patrick Walsh	91105649
Siu Leung	51390400
Tina Alinaghian	57291129
Fang Ming Lo	52013421
Austin Lee	50792270

Schedule of Leadership Resources:

The leadership rotation schedule of Team 10 is based on the schedule of the class. Since there are five deliverables, each team member is responsible for a deliverable and takes leadership of the group for that deliverable. The following is the leadership rotation schedule for each deliverable:

Deliverable	Due date	Leader
Requirements Iteration 1	July 11	Patrick Walsh
Requirements Iteration 2 / Test Plan iteration 1	July 19	Siu Leung
Test Plan Iteration2 / Design Iteration1	July 26	Tina Alinaghian
Design Iteration2 / Code -Iteration1	August 8	Fang Ming Lo
Code-Iteration2 (Final) + / All final deliverables	August 27	Austin Lee

Member Responsibilities:

Each member of our development group is responsible for the development of individual component of the project. The individual task includes the requirement specification, testing plan, design, and implementation of their component:

Name	Responsibility
Patrick Walsh	PHPNuke blocks generation and miscellany
Siu Leung	Page generation (including themes)
Tina Alinaghian	Category Management
Fang Ming Lo	Site look & feel management
Austin Lee	Contributor functions

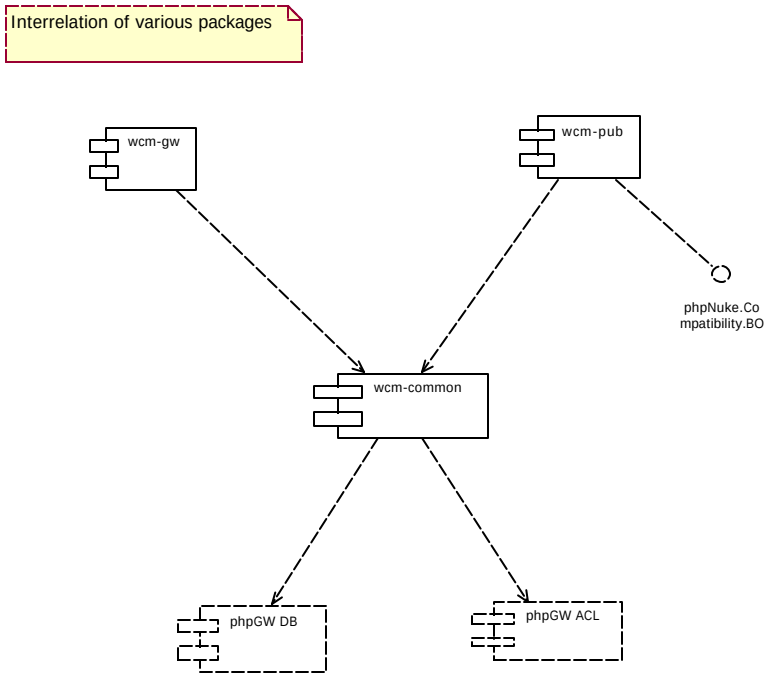
SECTION 2: ARCHITECTURAL OVERVIEW

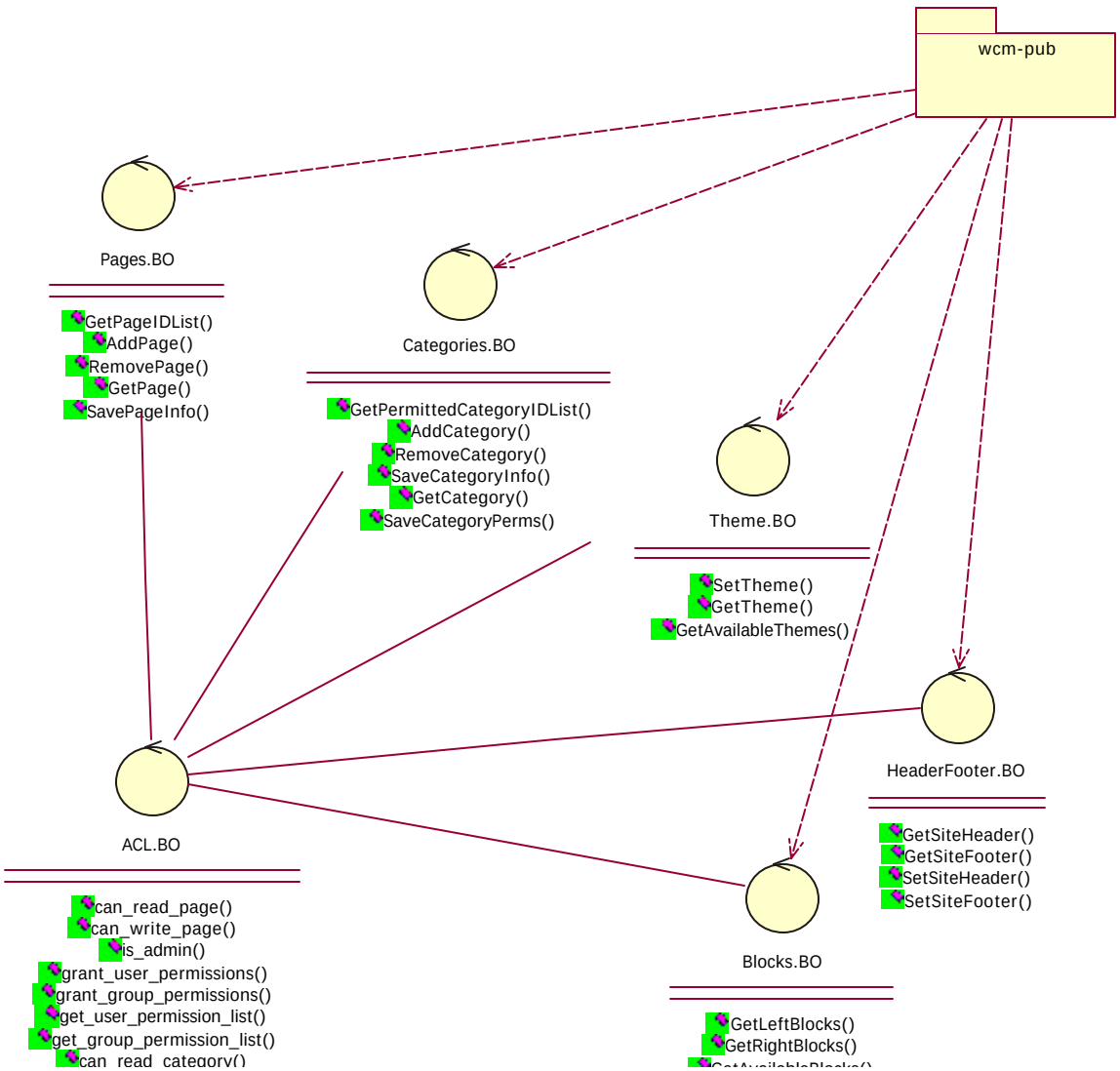
Architectural Style

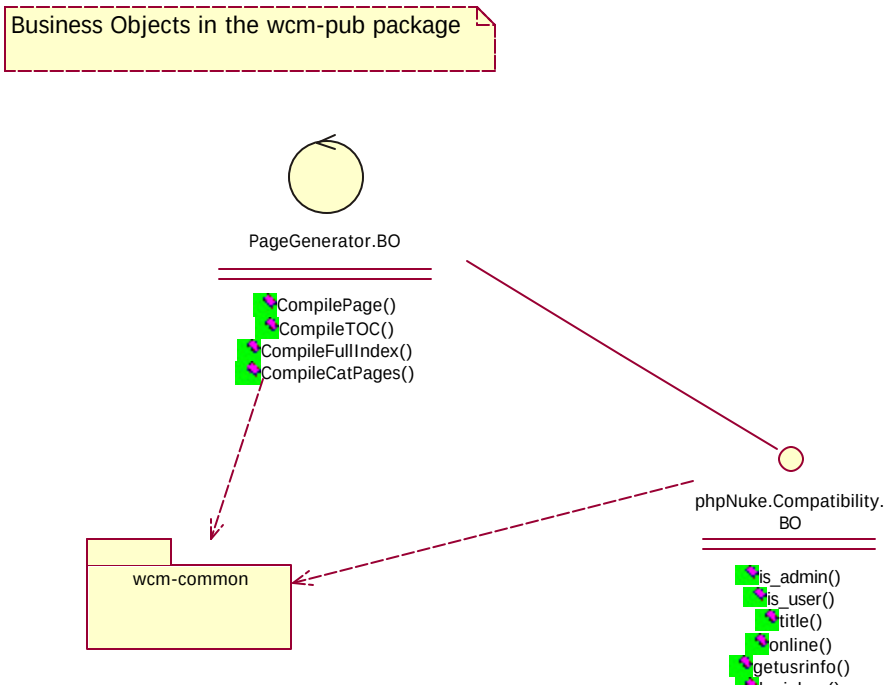
The architectural style used for the WCM application is pre-determined by the phpGroupWare project. As a phpGroupWare application, WCM will use a 3-tier, UI, BO, SO architecture. This architecture corresponds to the Boundary, Entity, Control class architecture espoused by Rational Rose. See the Design Overview for details. The application will consist of object oriented classes following distinctly into one of these three “tiers.”

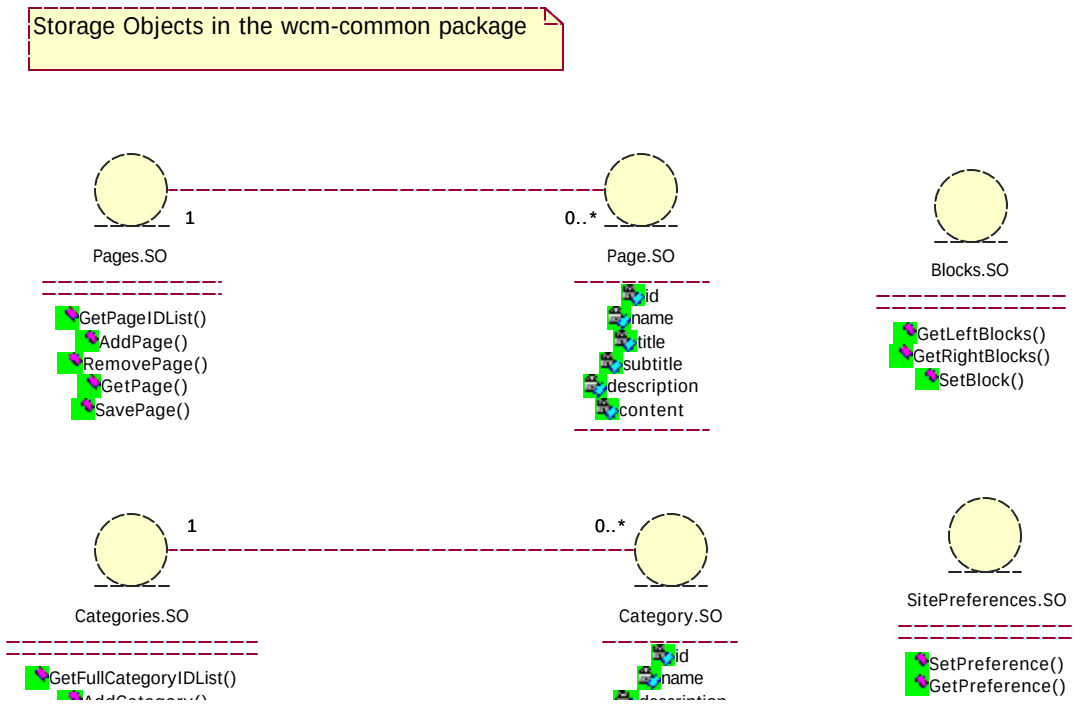
Additionally, the application has been separated into three packages. The two main packages represent the two disparate functions of the application: rendering the web site, and administering it. The web site is administered through the wcm-gw package. The web sit is rendered through the wcm-pub package. And the classes that are needed by both packages are in the wcm-common package.

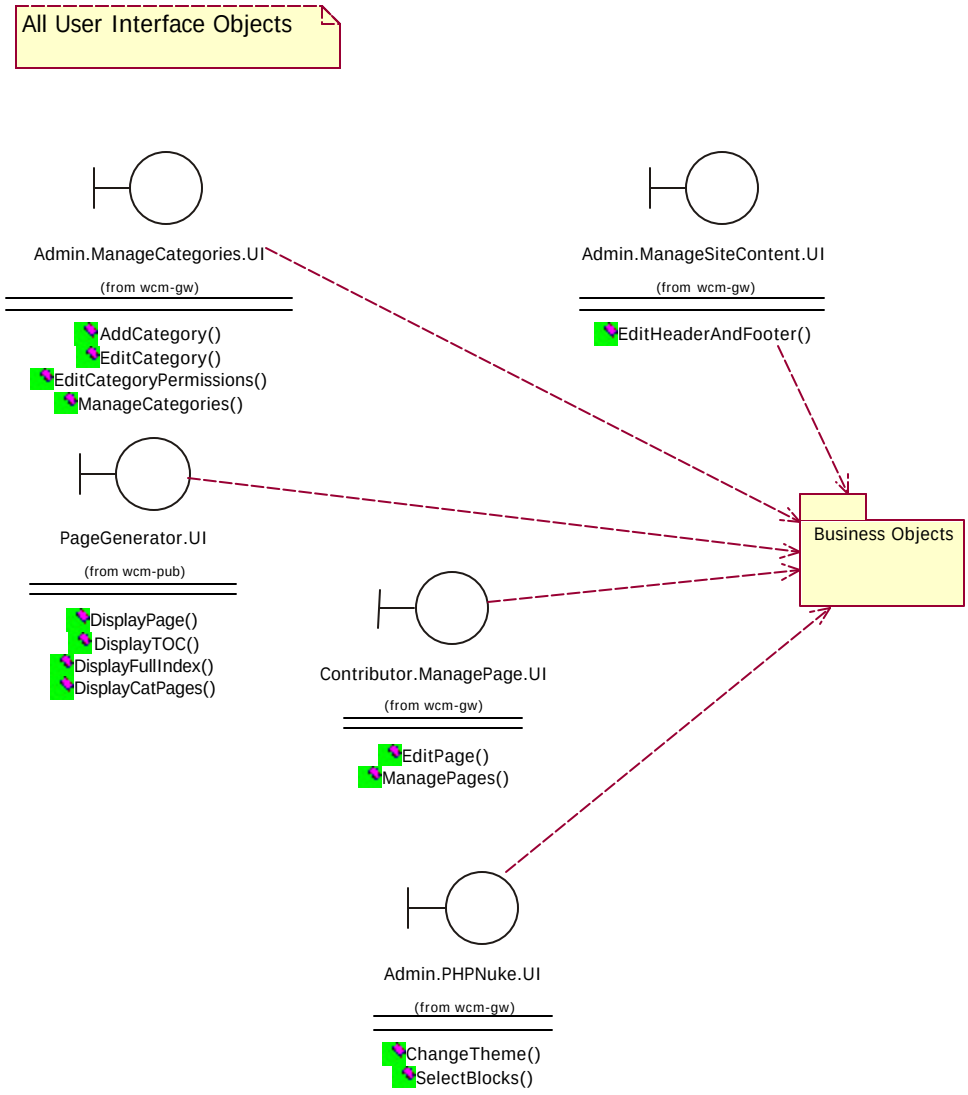
System Architecture Diagrams




Business Objects from the wcm-common package
wcm-pub
Pages.BO
GetPageIDList()
AddPage()
RemovePage()
GetPage()
SavePageInfo()
Categories.BO
GetPermittedCategoryIDList()
AddCategory()
RemoveCategory()
SaveCategoryInfo()
GetCategory()
SaveCategoryPerms()
Theme.BO
SetTheme()
GetTheme()
GetAvailableThemes()
HeaderFooter.BO
GetSiteHeader()
GetSiteFooter()
SetSiteHeader()
SetSiteFooter()
ACL.BO
can_read_page()
can_write_page()
is_admin()
grant_user_permissions()
grant_group_permissions()
get_user_permission_list()
get_group_permission_list()
Blocks.BO
GetLeftBlocks()
GetRightBlocks()


Business Objects in the wcm-pub package
PageGenerator.BO
CompilePage()
CompileTOC()
CompileFullIndex()
CompileCatPages()
wcm-common
phpNuke.Compatibility.
BO
is_admin()
is_user()
title()
online()
getusrinfo()


Storage Objects in the wcm-common package
1
0..*
Pages.SO
GetPageIDList()
AddPage()
RemovePage()
GetPage()
SavePage()
Page.SO
id
name
title
subtitle
description
content
Blocks.SO
GetLeftBlocks()
GetRightBlocks()
SetBlock()
1
0..*
Categories.SO
GetFullCategoryIDList()
Category.SO
id
name
SitePreferences.SO
SetPreference()
GetPreference()


All User Interface Objects
Admin.ManageCategories.UI
(from wcm-gw)
AddCategory()
EditCategory()
EditCategoryPermissions()
ManageCategories()
Admin.ManageSiteContent.UI
(from wcm-gw)
EditHeaderAndFooter()
PageGenerator.UI
(from wcm-pub)
DisplayPage()
DisplayTOC()
DisplayFullIndex()
DisplayCatPages()
Business Objects
Contributor.ManagePage.UI
(from wcm-gw)
EditPage()
ManagePages()
Admin.PHPNuke.UI
(from wcm-gw)
ChangeTheme()
SelectBlocks()

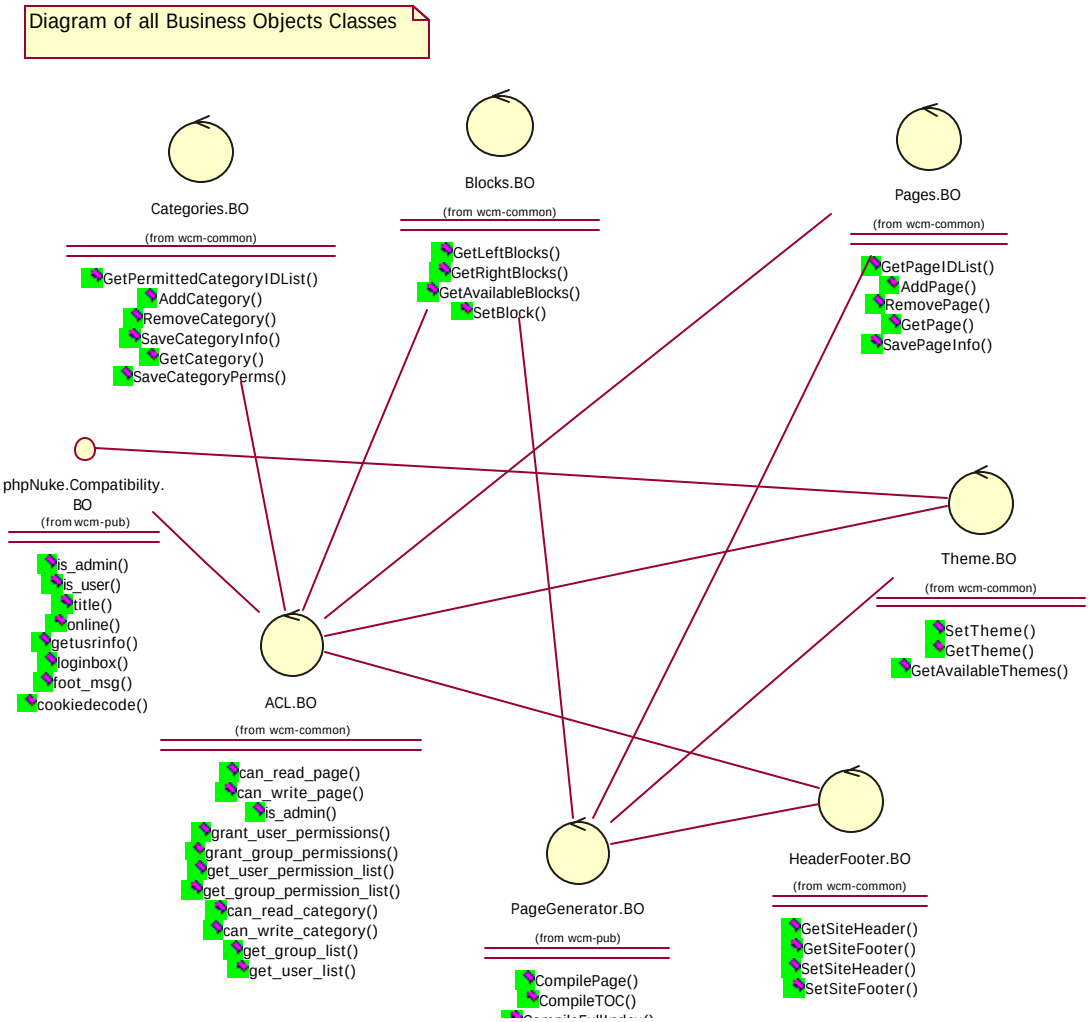
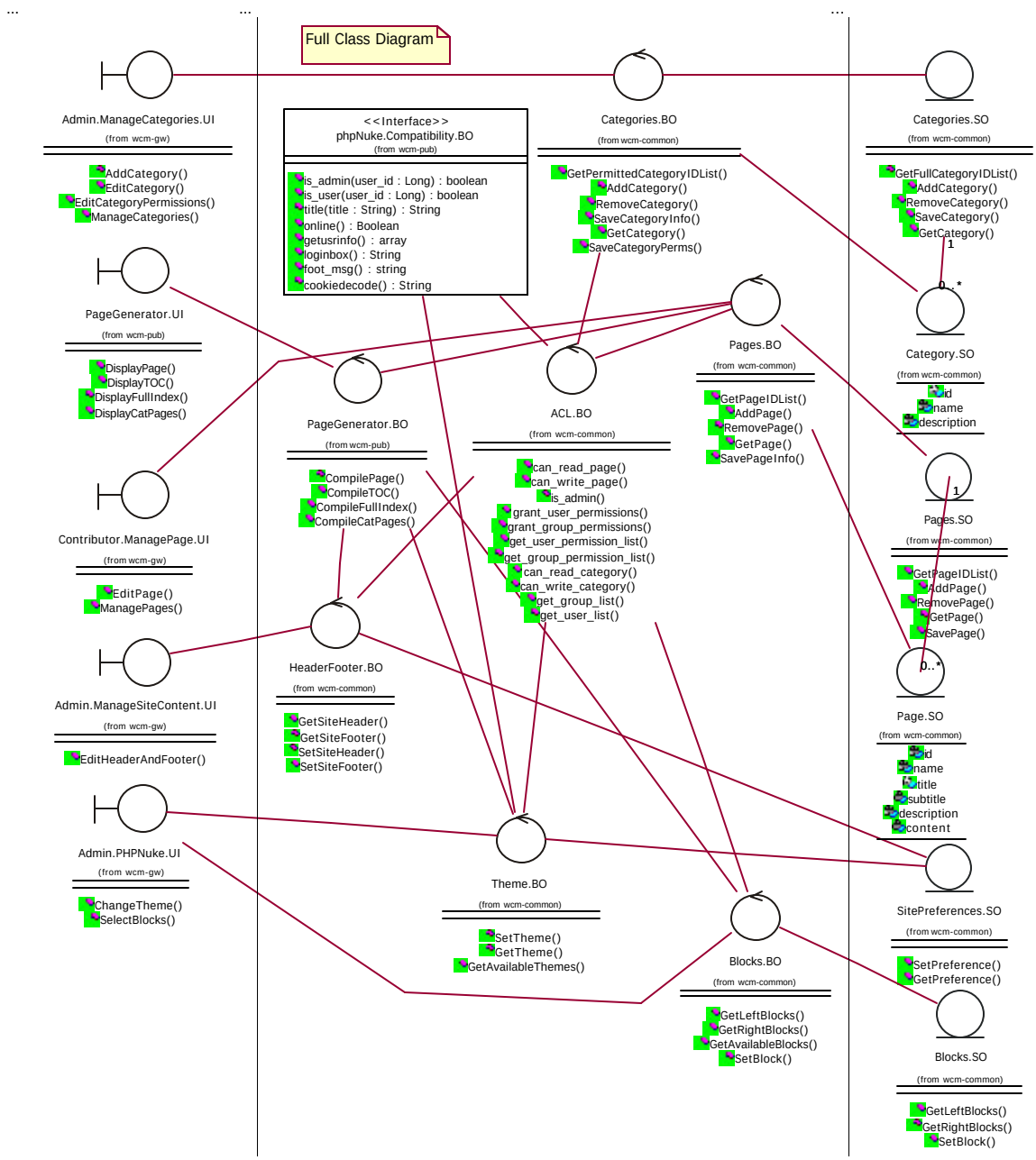


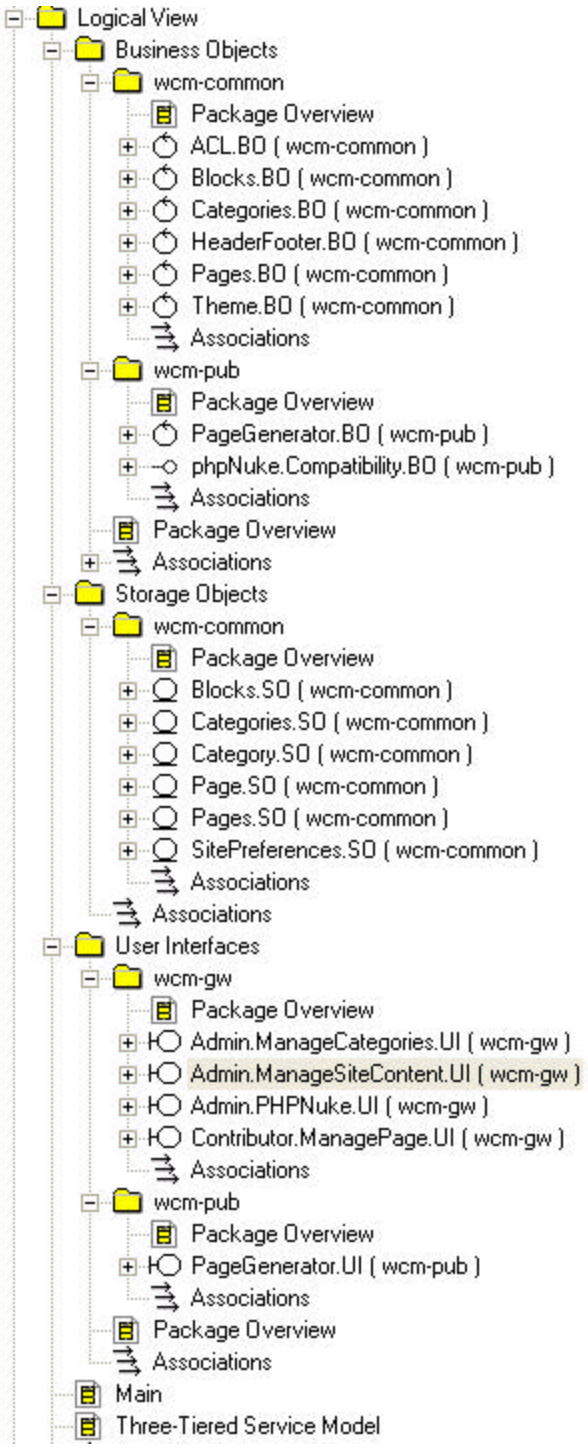
Diagram of all Business Objects Classes
Categories.BO
(from wcm-common)
GetPermittedCategoryIDList()
AddCategory()
RemoveCategory()
SaveCategoryInfo()
GetCategory()
SaveCategoryPerms()
Blocks.BO
(from wcm-common)
GetLeftBlocks()
GetRightBlocks()
GetAvailableBlocks()
SetBlock()
Pages.BO
(from wcm-common)
GetPageIDList()
AddPage()
RemovePage()
GetPage()
SavePageInfo()
phpNuke.Compatibility.BO
(from wcm-pub)
is_admin()
is_user()
title()
online()
getusrinfo()
loginbox()
foot_msg()
cookiedecode()
ACL.BO
(from wcm-common)
can_read_page()
can_write_page()
is_admin()
grant_user_permissions()
grant_group_permissions()
get_user_permission_list()
get_group_permission_list()
can_read_category()
can_write_category()
get_group_list()
get_user_list()
Theme.BO
(from wcm-common)
SetTheme()
GetTheme()
GetAvailableThemes()
PageGenerator.BO
(from wcm-pub)
CompilePage()
CompileTOC()
HeaderFooter.BO
(from wcm-common)
GetSiteHeader()
GetSiteFooter()
SetSiteHeader()
SetSiteFooter()



Subsystem Narrative

The foregoing diagrams illustrate the associations between classes, as well as the methods and properties of those classes. The diagrams have been organized by packages. Those packages without classes are, of course, not shown. For example, there are no User Interface classes in the wcm-common package. So there is no diagram showing the wcm-common User Interface classes. Likewise,

there are no Storage Object classes specific to either wcm-gw or wcm-pub, so these diagrams are non-existent. The shown diagrams comprehensively detail the packages and their relations. To help clarify the structure, a hierarchical view is shown below.



The classes shown in the hierarchy above are a screen shot from Rational Rose. Rational Rose is a fantastic program, but I have a few issues with it. The first is that there is no way to print out or export this hierarchical view, nor is there an easy way to print out selected specification information. Specifically, each class above has a list of functions or properties with parameters, descriptions, etc. The best I can do is print out a comprehensive list of the specifications. This is a textual representation of all of the data in the classes. It comes out to 171 pages. I'll put a sample here, but I suspect more than that would be overkill.

ACL.BO

Package: wcm-common
Stereotype: control
External Documents:
Export Control: Public
Cardinality: 1..1
Hierarchy:
Superclasses: none
Associations:
<no rolename> : Blocks.BO in association <unnamed>
<no rolename> : Categories.BO in association <unnamed>
<no rolename> : HeaderFooter.BO in association <unnamed>
<no rolename> : Pages.BO in association <unnamed>
<no rolename> : phpNuke.Compatibility.BO in association <unnamed>
<no rolename> : Theme.BO in association <unnamed>
Public Interface:
Operations:
can_read_page
can_write_page
is_admin
grant_user_permissions
grant_group_permissions
get_user_permission_list
get_group_permission_list
can_read_category
can_write_category
get_group_list
get_user_list
State machine: No
Concurrency: Sequential
Persistence: Transient
Operation name:

can_read_page

Public member of: ACL.BO
Return Class: Boolean
Arguments:
long page
Concurrency: Sequential
State machine: No
Operation name:

can_write_page

Public member of: ACL.BO
Return Class: Boolean
Arguments:
long page
Concurrency: Sequential
State machine: No
File: C:\Documents and Settings\pwalsh\Desktop\wcm\wcm-interfaces.mdl 2:05:23 AM Thursday, August 08, 2002 Page 3
Operation name:

is_admin

Public member of: ACL.BO
Return Class: Boolean
Concurrency: Sequential
State machine: No
Operation name:

grant_user_permissions

Public member of: ACL.BO
Return Class: Boolean
Arguments:
Long user
Long page
Boolean can_read
Boolean can_write
Concurrency: Sequential
State machine: No
Operation name:

grant_group_permissions

Public member of: ACL.BO
Return Class: Boolean
Arguments:
group
page
can_read
can_write
Concurrency: Sequential
State machine: No
Operation name:

get_user_permission_list

Public member of: ACL.BO
Return Class: Array
Concurrency: Sequential
State machine: No
Operation name:

get_group_permission_list

Public member of: ACL.BO
File: C:\Documents and Settings\pwalsh\Desktop\wcm\wcm-interfaces.mdl 2:05:23 AM Thursday, August 08, 2002 Page 4
Return Class: Array
Concurrency: Sequential
State machine: No
Operation name:

can_read_category

Public member of: ACL.BO
Return Class: Boolean
Arguments:
Long category_id
Concurrency: Sequential
State machine: No
Operation name:

can_write_category

Public member of: ACL.BO
Return Class: Boolean
Arguments:
Long category_id
Concurrency: Sequential
State machine: No
Operation name:

get_group_list

Public member of: ACL.BO
Return Class: Array
Concurrency: Sequential
State machine: No
Operation name:

get_user_list

Public member of: ACL.BO
Return Class: Array
Concurrency: Sequential
State machine: No
Class name:

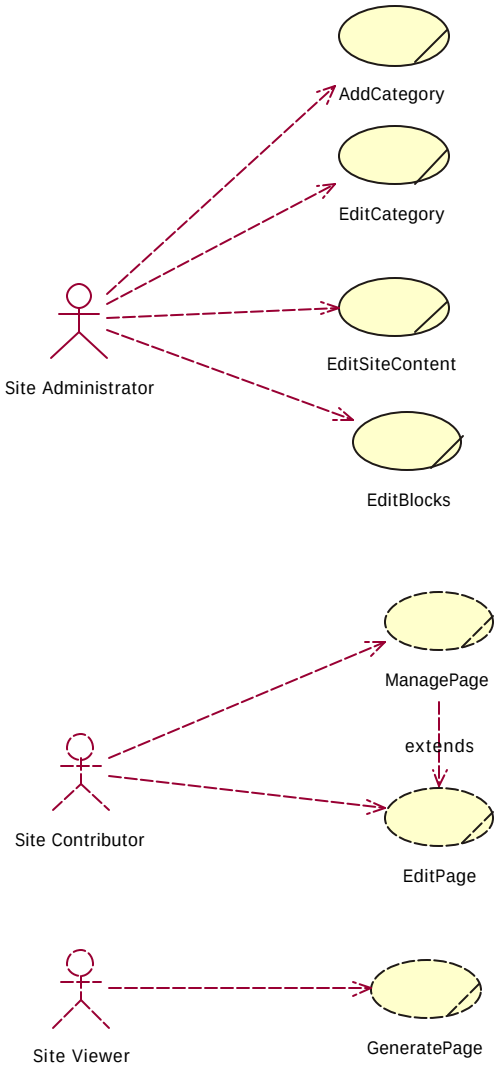
Limitations of Design

The limitations of the design are not yet obvious. Certainly there are some built-in limitations by working under the phpGroupWare platform – for example, javascript and php4 only functions are not allowed. But that is more a limitation of the requirements than of the design. The limitations will become obvious when the program is finally being used and feature requests start coming in.

SECTION 3: USE CASE REALIZATIONS

Use Cases

Well, if you've made it this far and you're still attentive, my hat is off to you. You're not even halfway through the document yet, though.



AddCategory Use Case

Contents:

1) AddCategory Use Case Realization
   a) Sequence Diagrams
      i) Basic Course

ii) Alternate Course 1

iii) Alternate Course 2

b) Classes

i) Boundary

(1) Admin.ManageCategories.UI

ii) Control

(1) Categories.BO

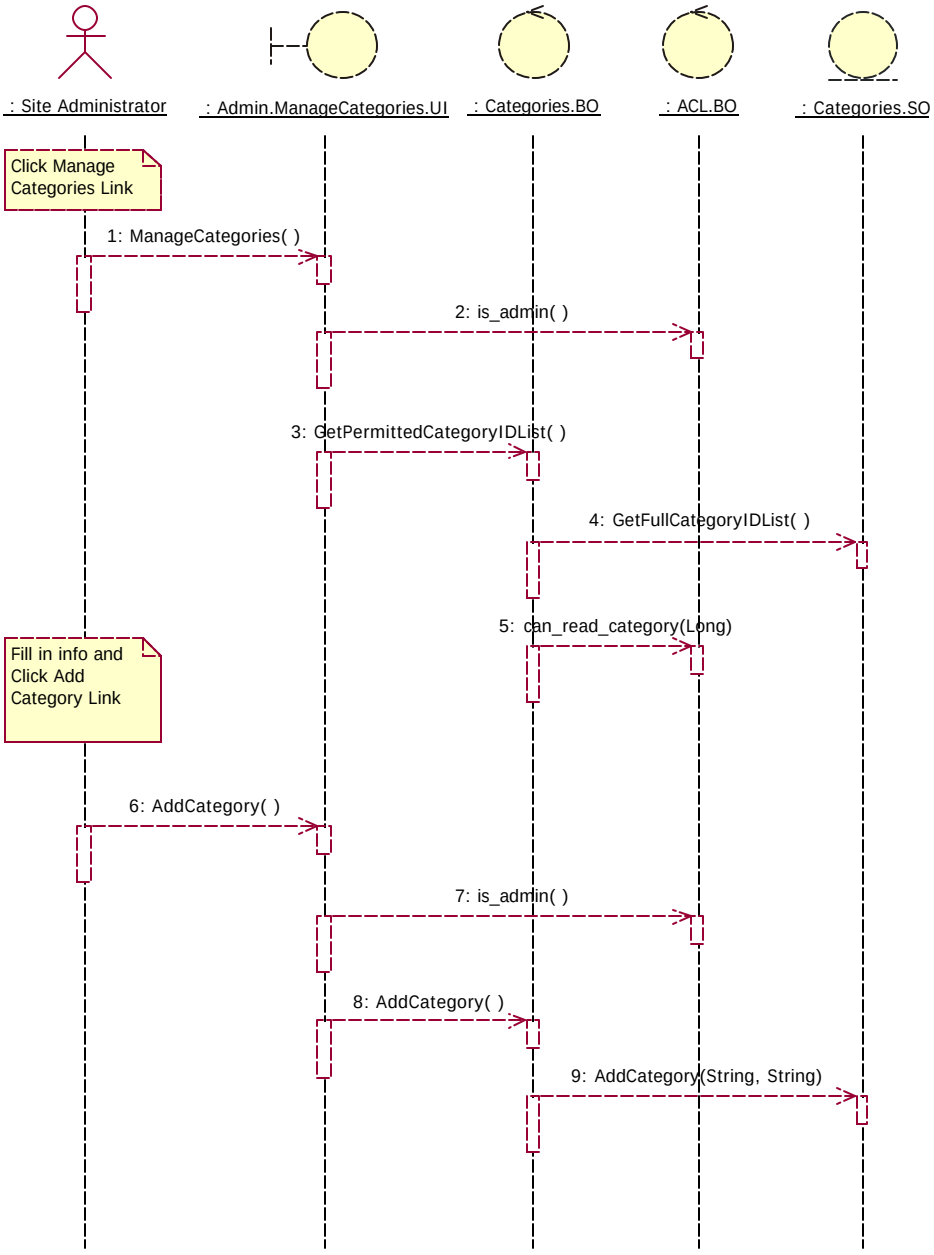
(2) ACL.BO

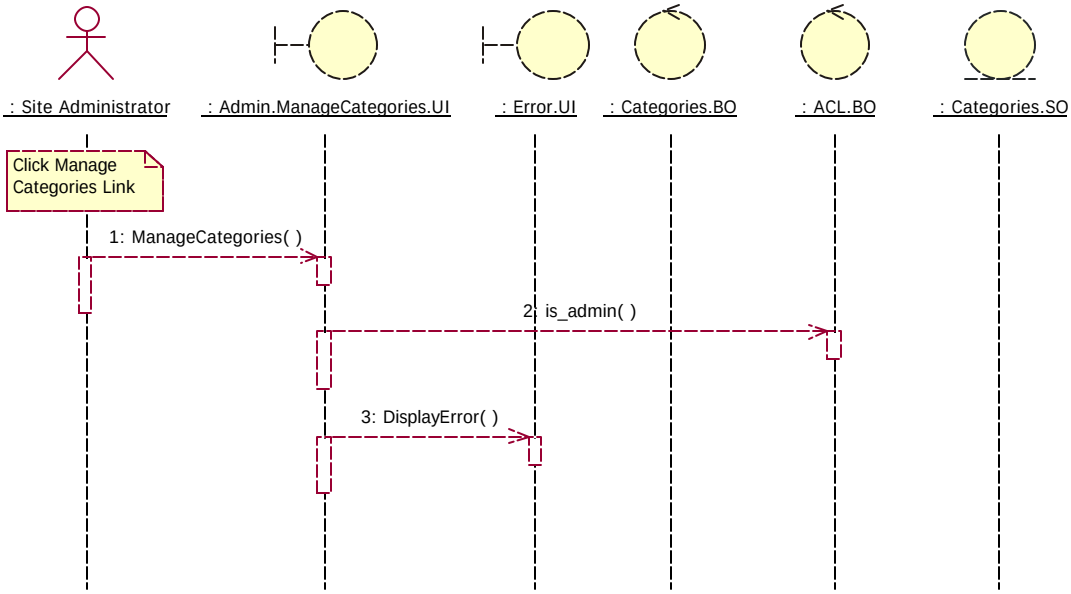
iii) Entity

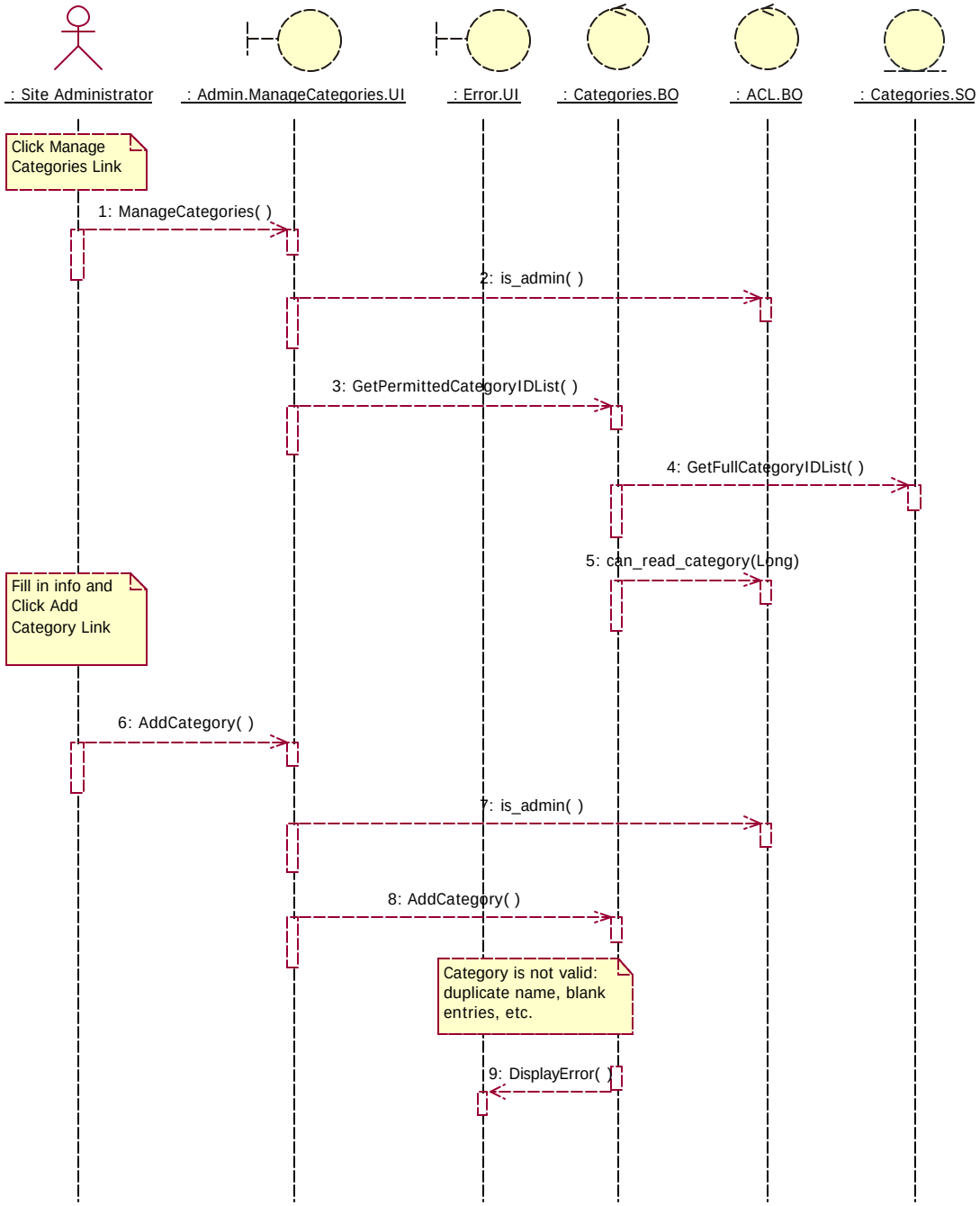
(1) Categories.SO

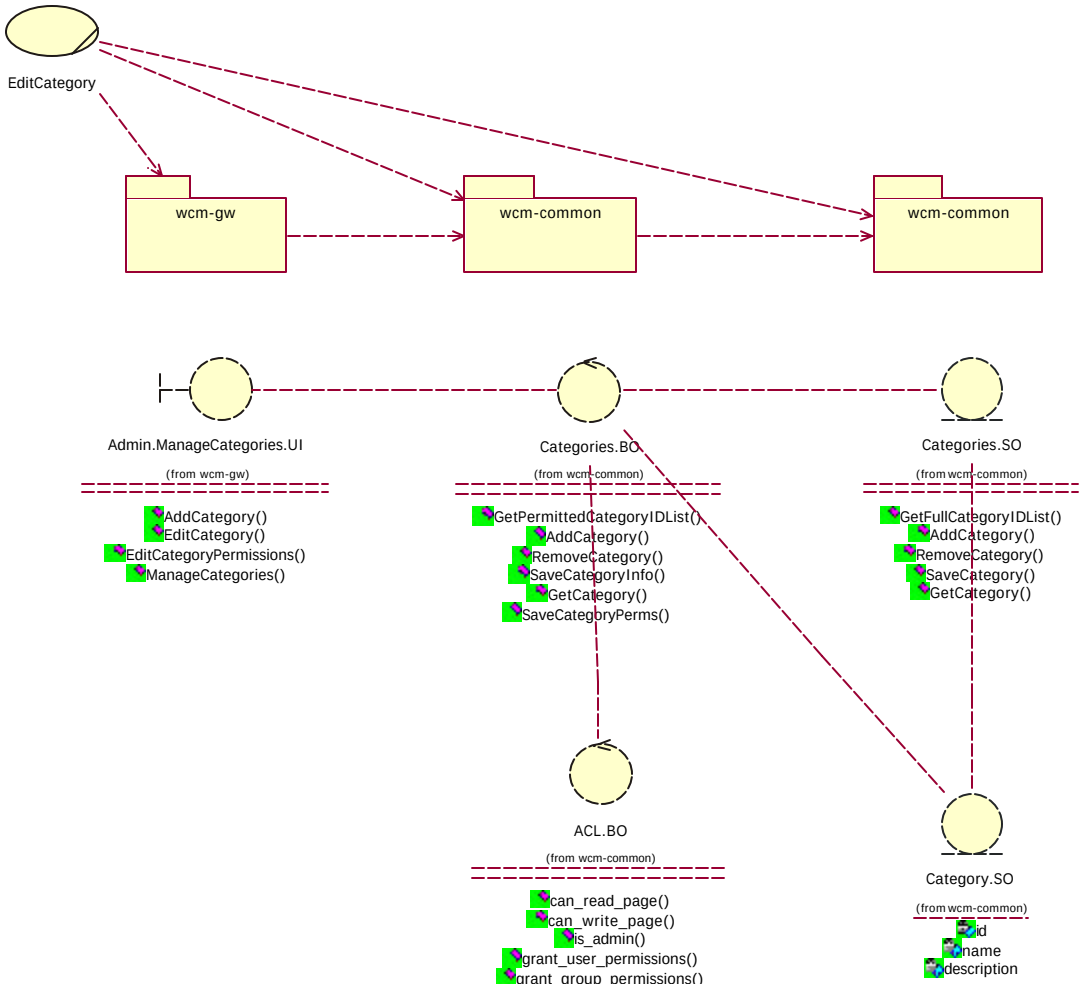
(2) Category.SO

Diagrams:


: Site Administrator
: Admin.ManageCategories.UI
: Categories.BO
: ACL.BO
: Categories.SO
Click Manage Categories Link
1: ManageCategories( )
2: is_admin( )
3: GetPermittedCategoryIDList( )
4: GetFullCategoryIDList( )
5: can_read_category(Long)
Fill in info and Click Add Category Link
6: AddCategory( )
7: is_admin( )
8: AddCategory( )
9: AddCategory(String, String)


: Site Administrator
: Admin.ManageCategories.UI
: Error.UI
: Categories.BO
: ACL.BO
: Categories.SO
Click Manage Categories Link
1: ManageCategories( )
2: is_admin( )
3: DisplayError( )


: Site Administrator
: Admin.ManageCategories.UI
: Error.UI
: Categories.BO
: ACL.BO
: Categories.SO
Click Manage Categories Link
1: ManageCategories( )
2: is_admin( )
3: GetPermittedCategoryIDList( )
4: GetFullCategoryIDList( )
5: can_read_category(Long)
Fill in info and Click Add Category Link
6: AddCategory( )
7: is_admin( )
8: AddCategory( )
Category is not valid: duplicate name, blank entries, etc.
9: DisplayError( )



EditCategory Use Case

Contents:

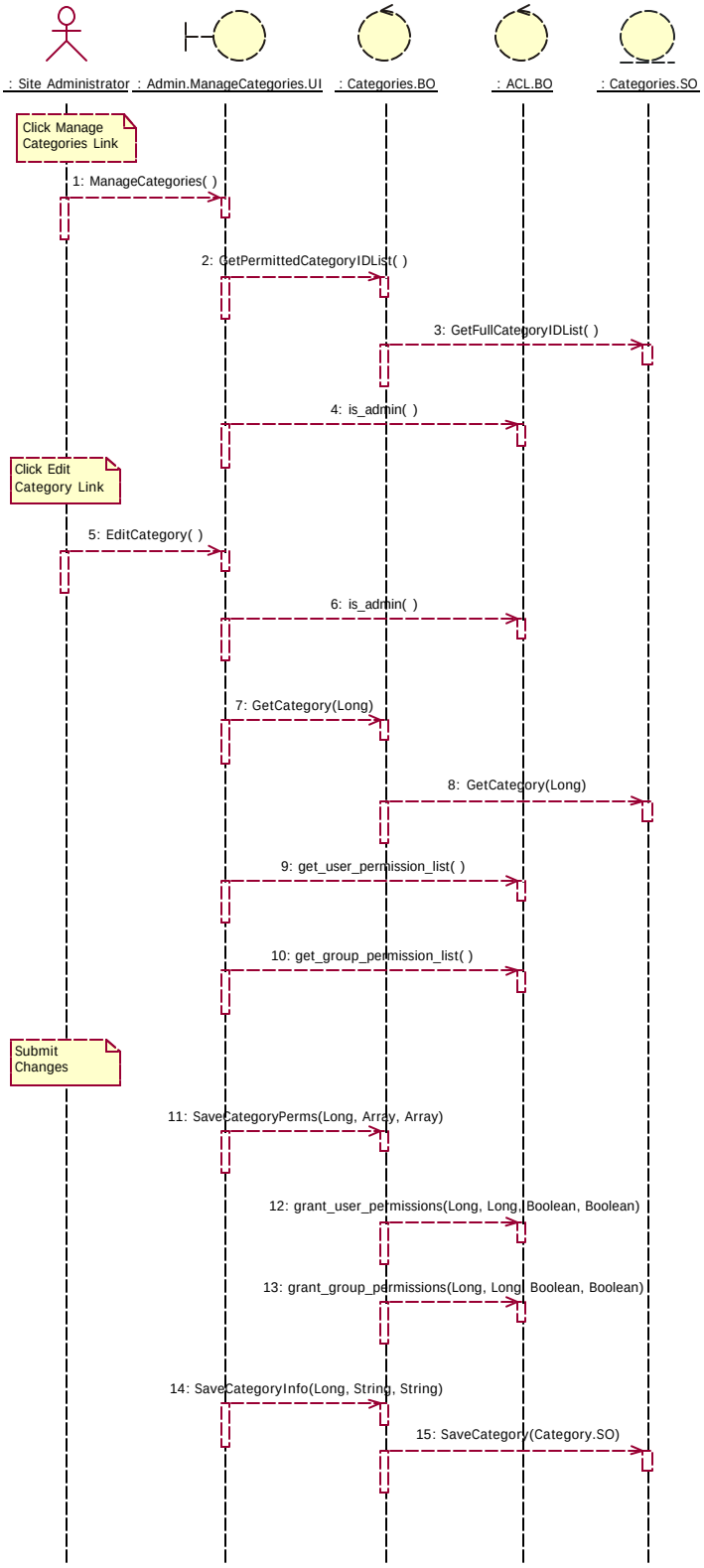
(2) ACL.BO

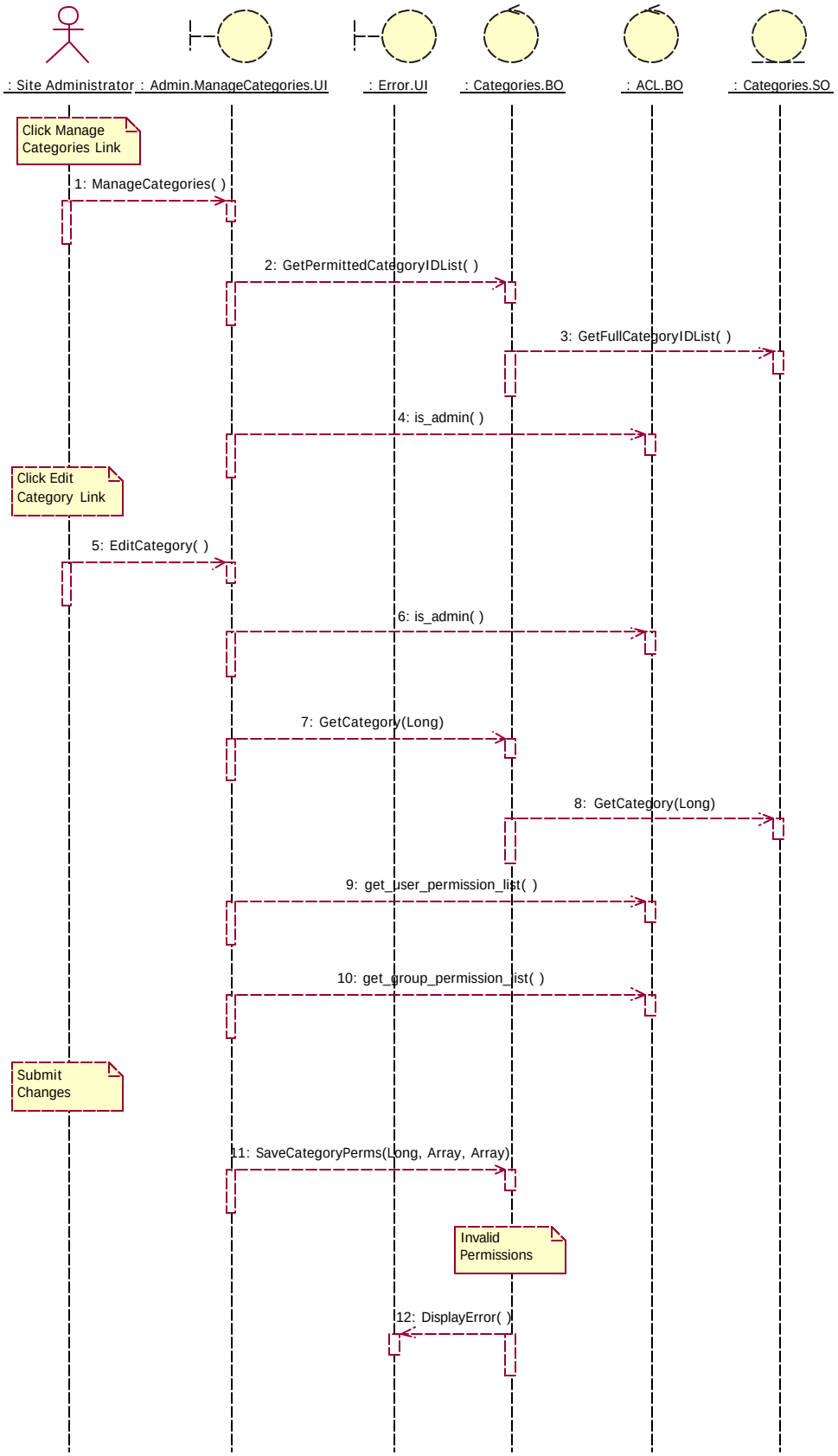
iii) Entity

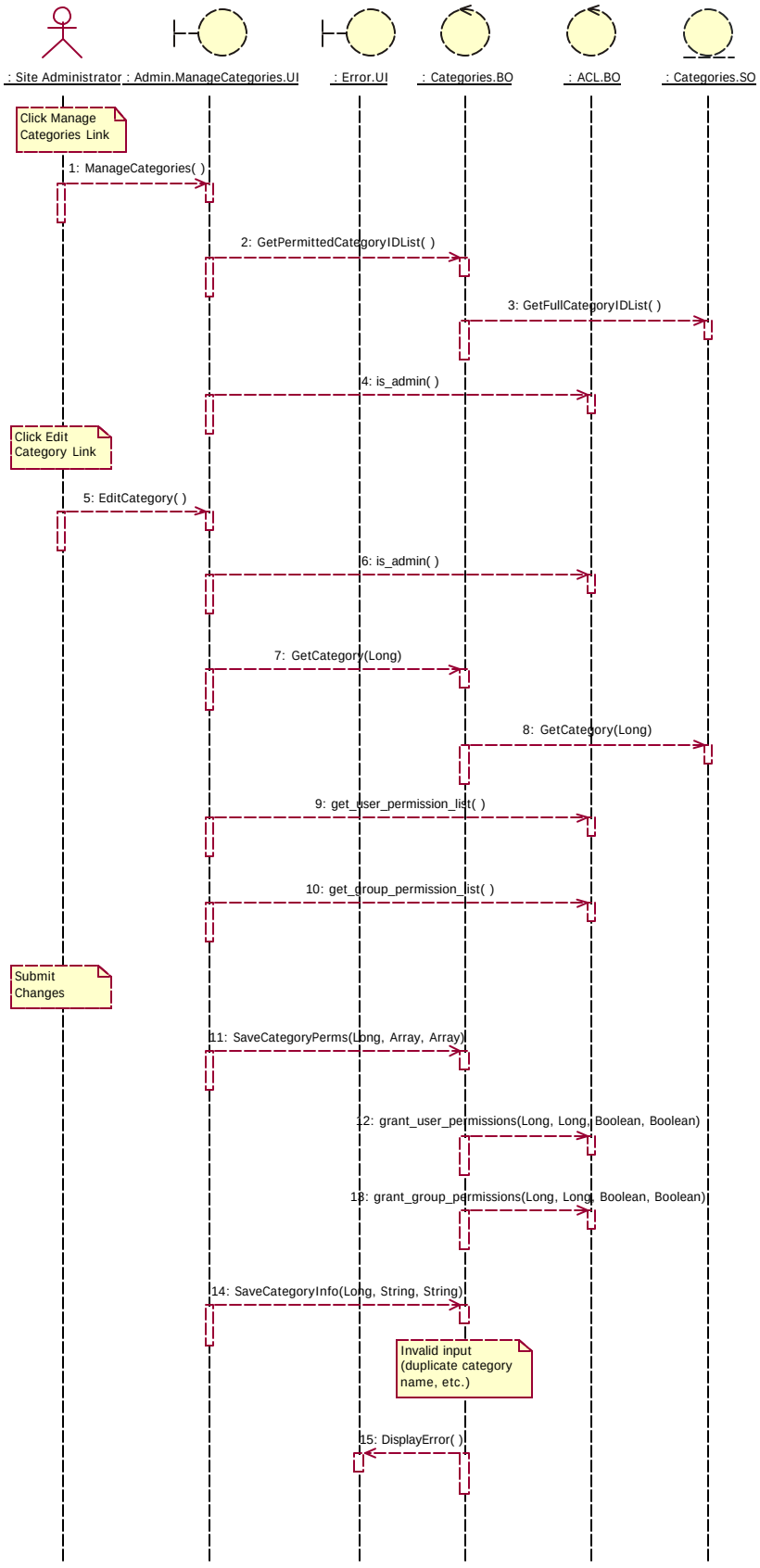
(1) Categories.SO

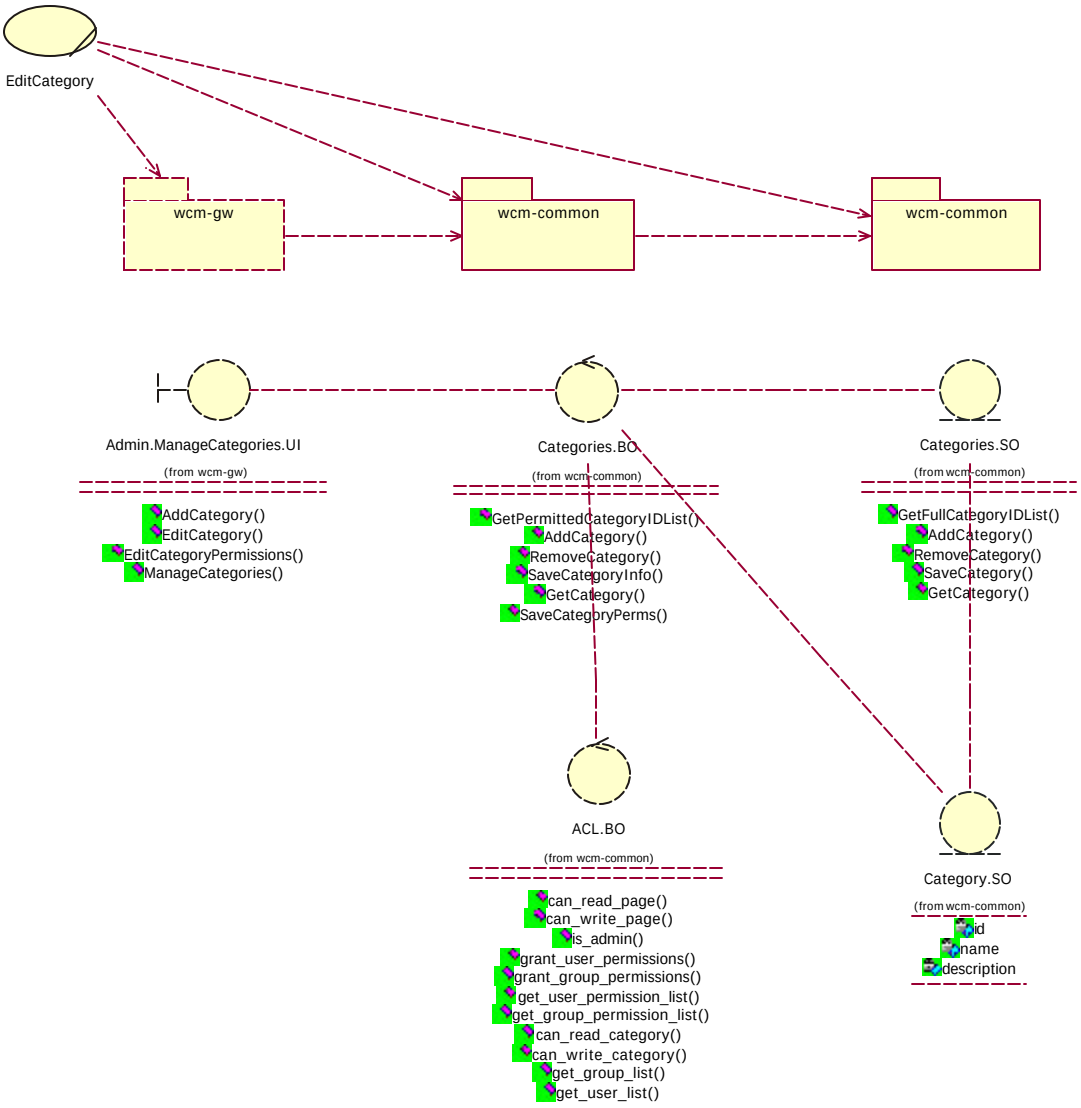
(2) Category.SO

Diagrams:


: Site Administrator
: Admin.ManageCategories.UI
: Categories.BO
: ACL.BO
: Categories.SO
Click Manage Categories Link
1: ManageCategories( )
2: GetPermittedCategoryIDList( )
3: GetFullCategoryIDList( )
4: is_admin( )
Click Edit Category Link
5: EditCategory( )
6: is_admin( )
7: GetCategory(Long)
8: GetCategory(Long)
9: get_user_permission_list( )
10: get_group_permission_list( )
Submit Changes
11: SaveCategoryPerms(Long, Array, Array)
12: grant_user_permissions(Long, Long, Boolean, Boolean)
13: grant_group_permissions(Long, Long, Boolean, Boolean)
14: SaveCategoryInfo(Long, String, String)
15: SaveCategory(Category.SO)


: Site Administrator
: Admin.ManageCategories.UI
: Error.UI
: Categories.BO
: ACL.BO
: Categories.SO
Click Manage Categories Link
1: ManageCategories( )
2: GetPermittedCategoryIDList( )
3: GetFullCategoryIDList( )
4: is_admin( )
Click Edit Category Link
5: EditCategory( )
6: is_admin( )
7: GetCategory(Long)
8: GetCategory(Long)
9: get_user_permission_list( )
10: get_group_permission_list( )
Submit Changes
11: SaveCategoryPerms(Long, Array, Array)
Invalid Permissions
12: DisplayError( )


: Site Administrator
: Admin.ManageCategories.UI
: Error.UI
: Categories.BO
: ACL.BO
: Categories.SO
Click Manage Categories Link
1: ManageCategories( )
2: GetPermittedCategoryIDList( )
3: GetFullCategoryIDList( )
4: is_admin( )
Click Edit Category Link
5: EditCategory( )
6: is_admin( )
7: GetCategory(Long)
8: GetCategory(Long)
9: get_user_permission_list( )
10: get_group_permission_list( )
Submit Changes
11: SaveCategoryPerms(Long, Array, Array)
12: grant_user_permissions(Long, Long, Boolean, Boolean)
13: grant_group_permissions(Long, Long, Boolean, Boolean)
14: SaveCategoryInfo(Long, String, String)
Invalid input (duplicate category name, etc.)
15: DisplayError( )



EditSite Content Use Case

Contents:

1) EditSite Use Case Realization
   a) Sequence Diagrams
      i) Basic Course
      ii) Alternate Course 1
      iii) Alternate Course 2

b) Classes

i) Boundary

(1) Admin.ManageSiteContent.UI

ii) Control

(1) HeaderFooter.BO

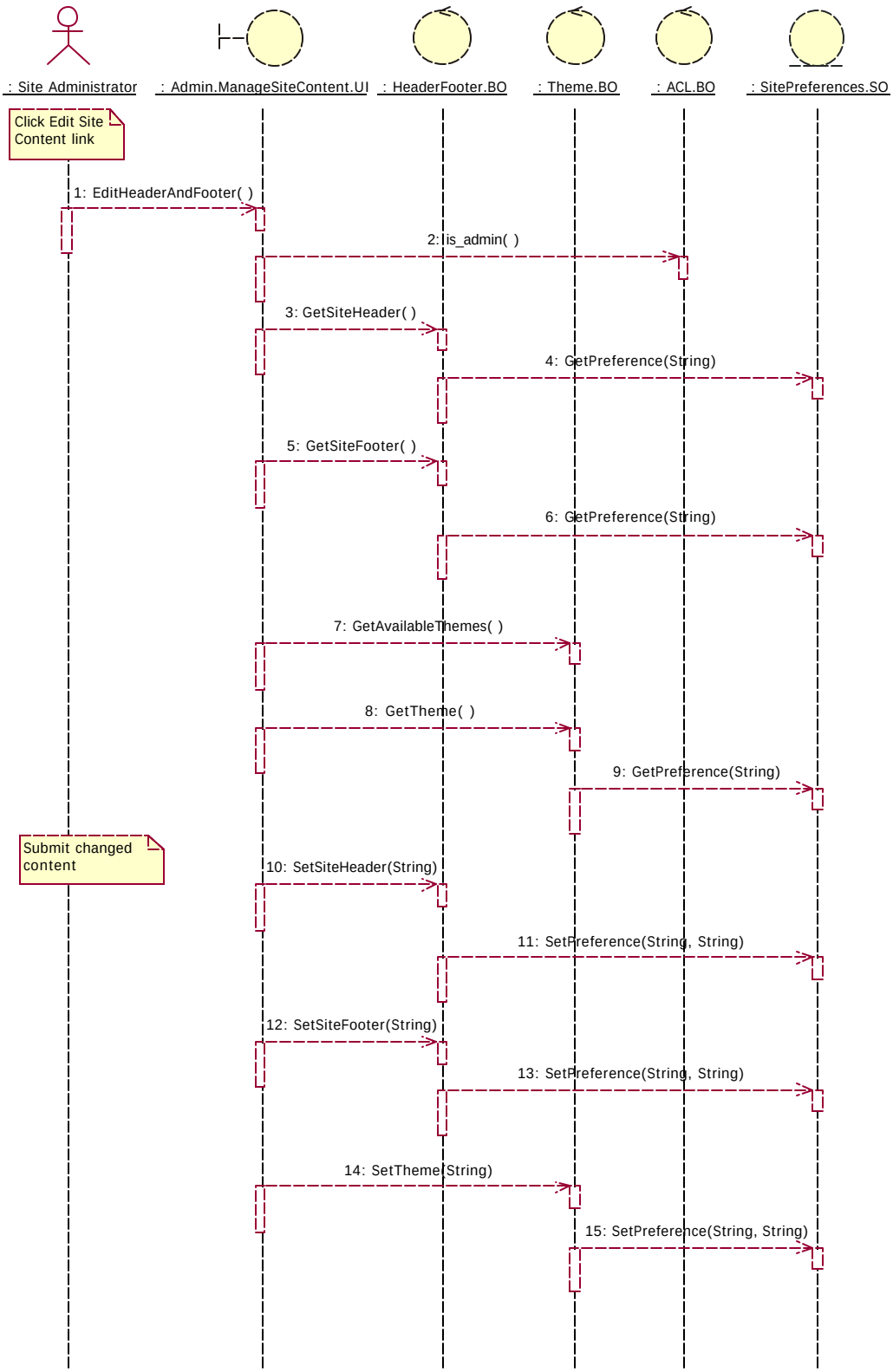
(2) Theme.BO

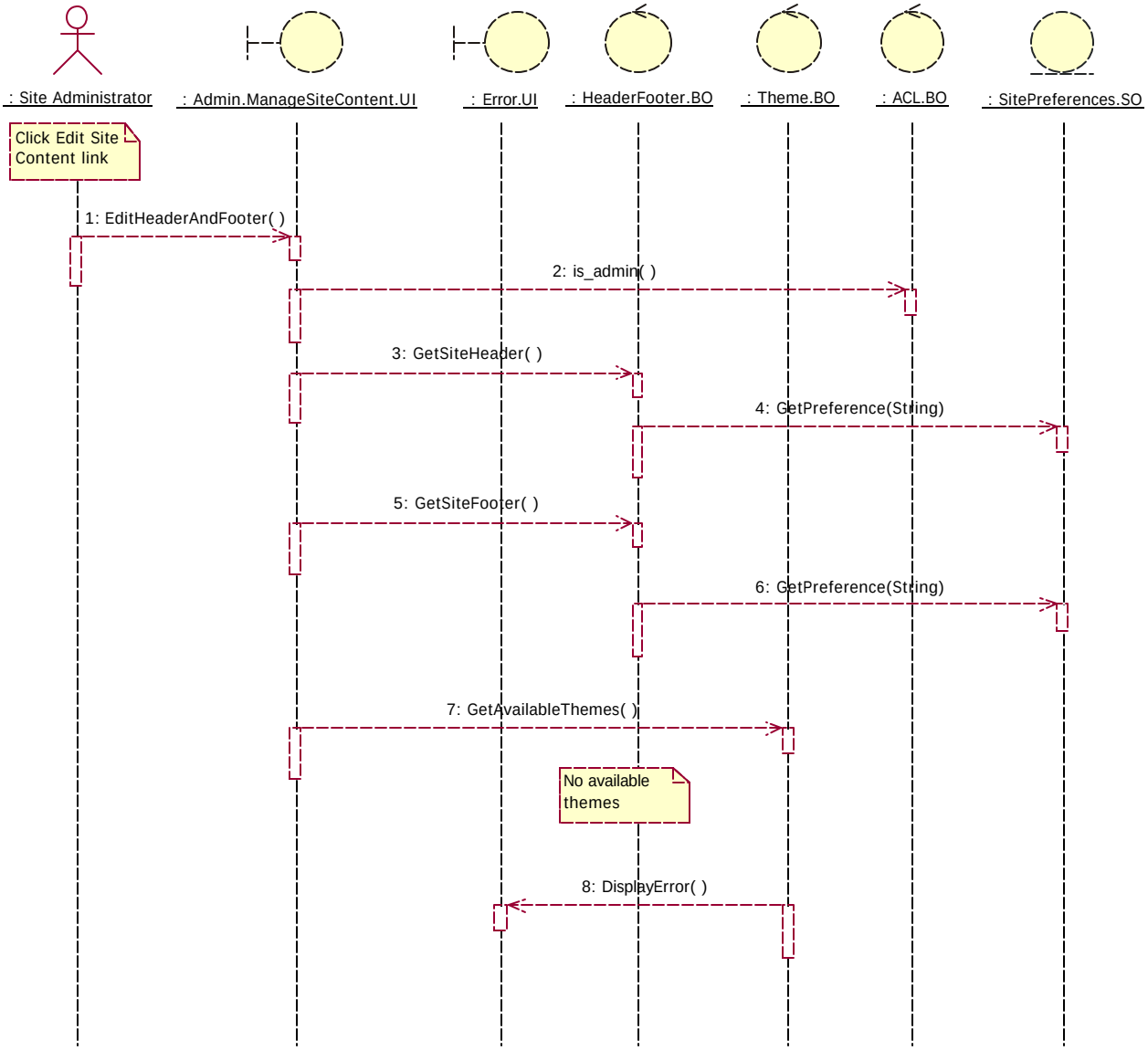
(3) ACL.BO

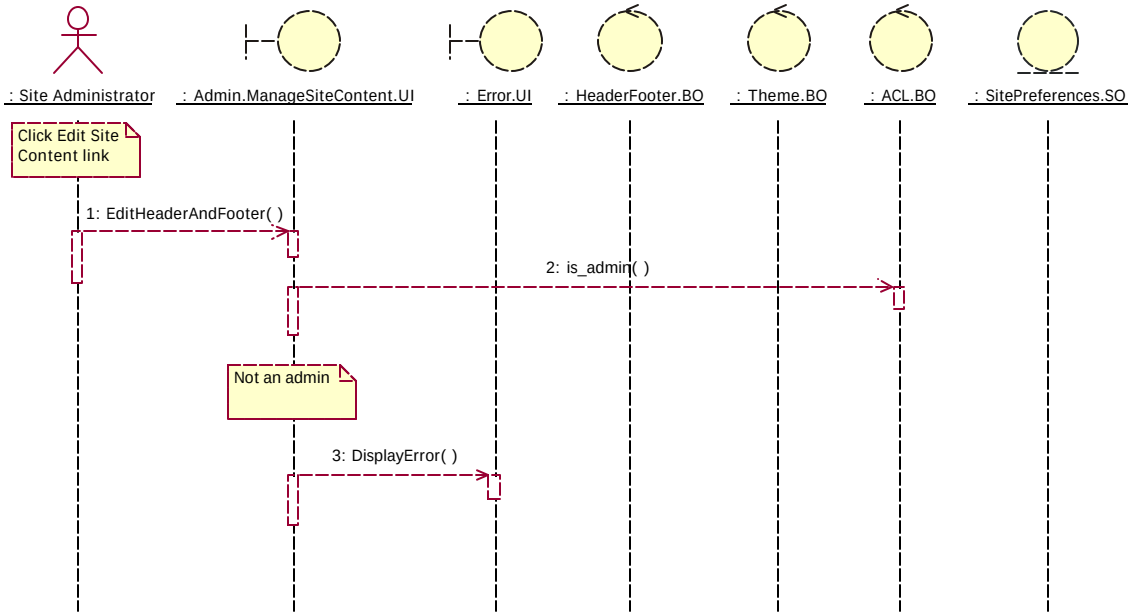
iii) Entity

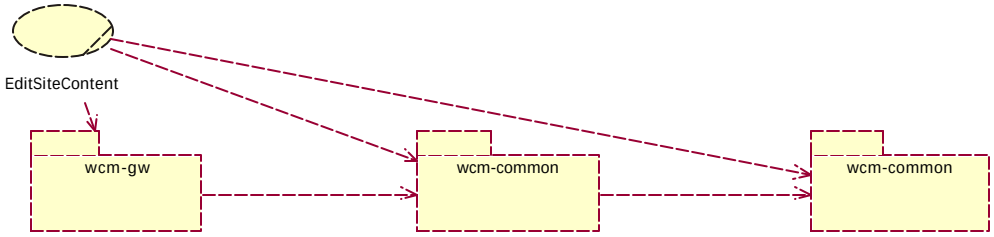
(1) SitePreferences.SO

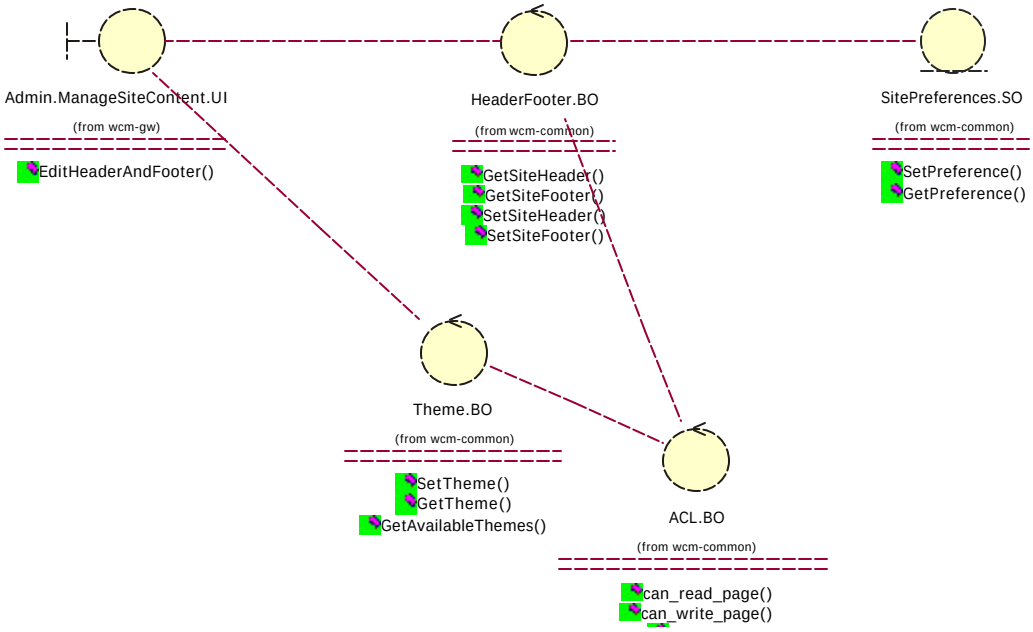
Diagrams:


: Site Administrator
: Admin.ManageSiteContent.UI
: HeaderFooter.BO
: Theme.BO
: ACL.BO
: SitePreferences.SO
Click Edit Site Content link
1: EditHeaderAndFooter( )
2: is_admin( )
3: GetSiteHeader( )
4: GetPreference(String)
5: GetSiteFooter( )
6: GetPreference(String)
7: GetAvailableThemes( )
8: GetTheme( )
9: GetPreference(String)
Submit changed content
10: SetSiteHeader(String)
11: SetPreference(String, String)
12: SetSiteFooter(String)
13: SetPreference(String, String)
14: SetTheme(String)
15: SetPreference(String, String)


: Site Administrator
: Admin.ManageSiteContent.UI
: Error.UI
: HeaderFooter.BO
: Theme.BO
: ACL.BO
: SitePreferences.SO
Click Edit Site Content link
1: EditHeaderAndFooter( )
2: is_admin( )
3: GetSiteHeader( )
4: GetPreference(String)
5: GetSiteFooter( )
6: GetPreference(String)
7: GetAvailableThemes( )
No available themes
8: DisplayError( )


: Site Administrator
: Admin.ManageSiteContent.UI
: Error.UI
: HeaderFooter.BO
: Theme.BO
: ACL.BO
: SitePreferences.SO
Click Edit Site Content link
1: EditHeaderAndFooter( )
2: is_admin( )
Not an admin
3: DisplayError( )

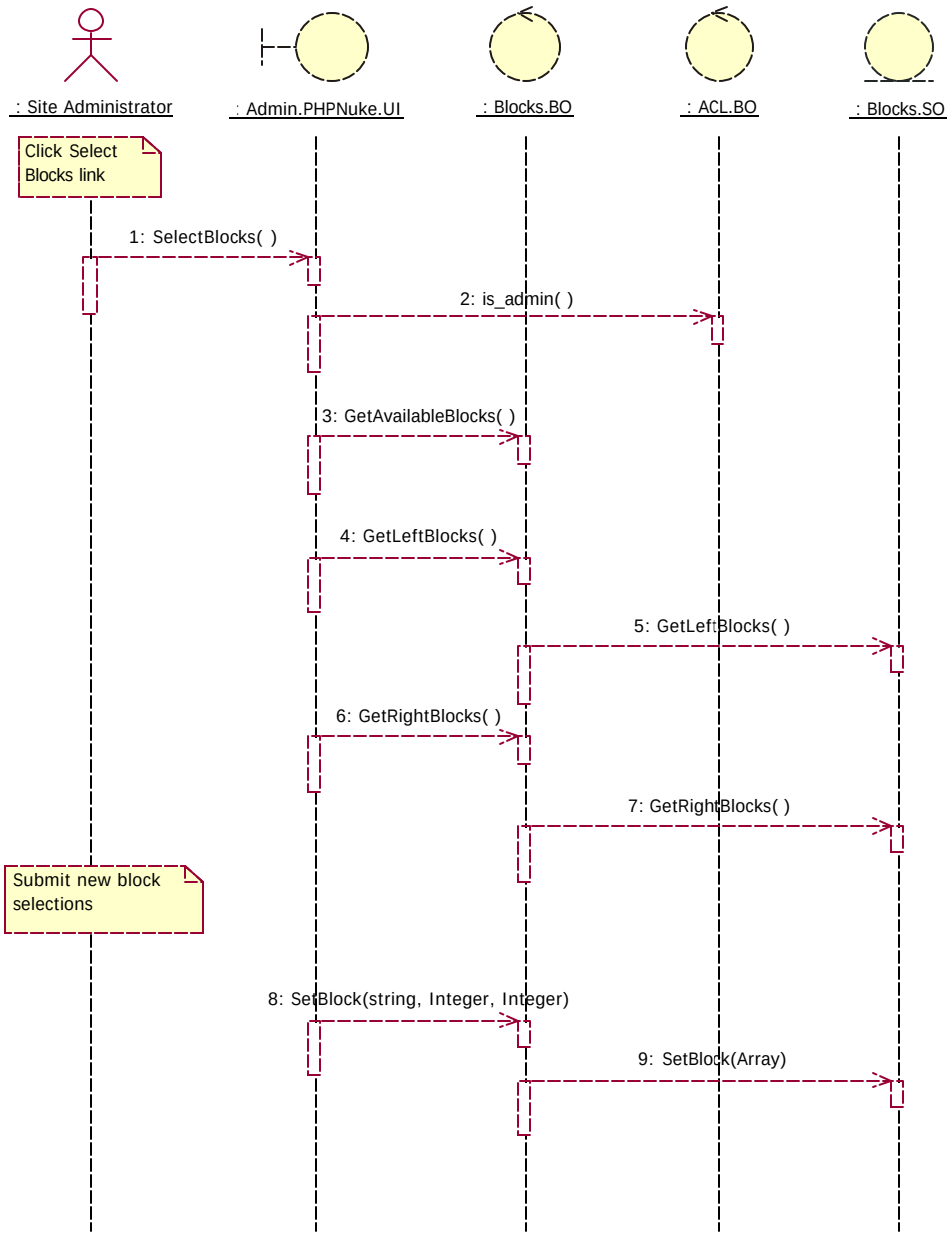

EditSiteContent
wcm-gw
wcm-common
wcm-common

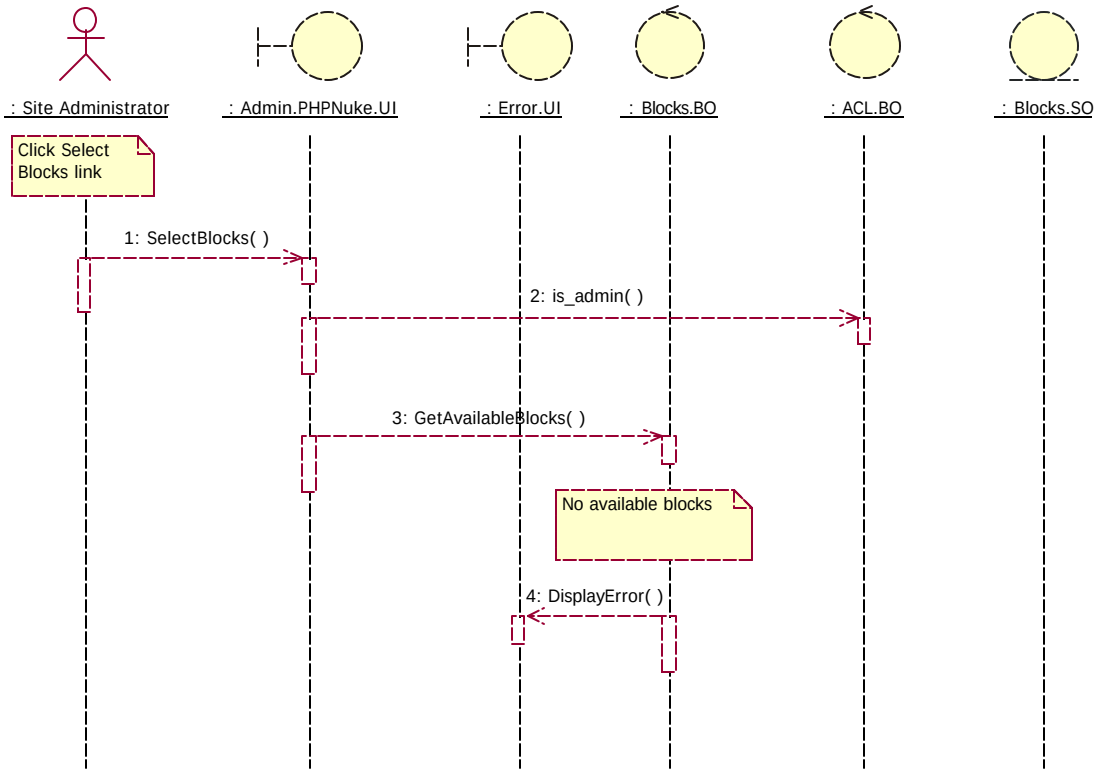

Admin.ManageSiteContent.UI
(from wcm-gw)
EditHeaderAndFooter()
HeaderFooter.BO
(from wcm-common)
GetSiteHeader()
GetSiteFooter()
SetSiteHeader()
SetSiteFooter()
SitePreferences.SO
(from wcm-common)
SetPreference()
GetPreference()
Theme.BO
(from wcm-common)
SetTheme()
GetTheme()
GetAvailableThemes()
ACL.BO
(from wcm-common)
can_read_page()
can_write_page()

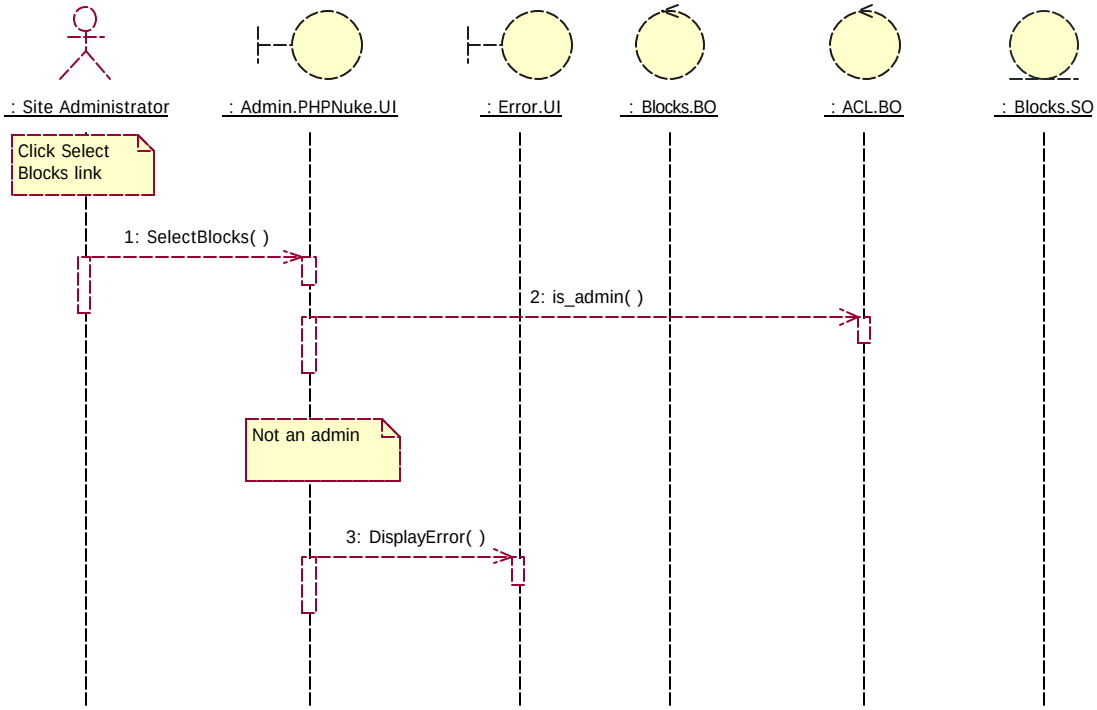
EditBlocks Use Case

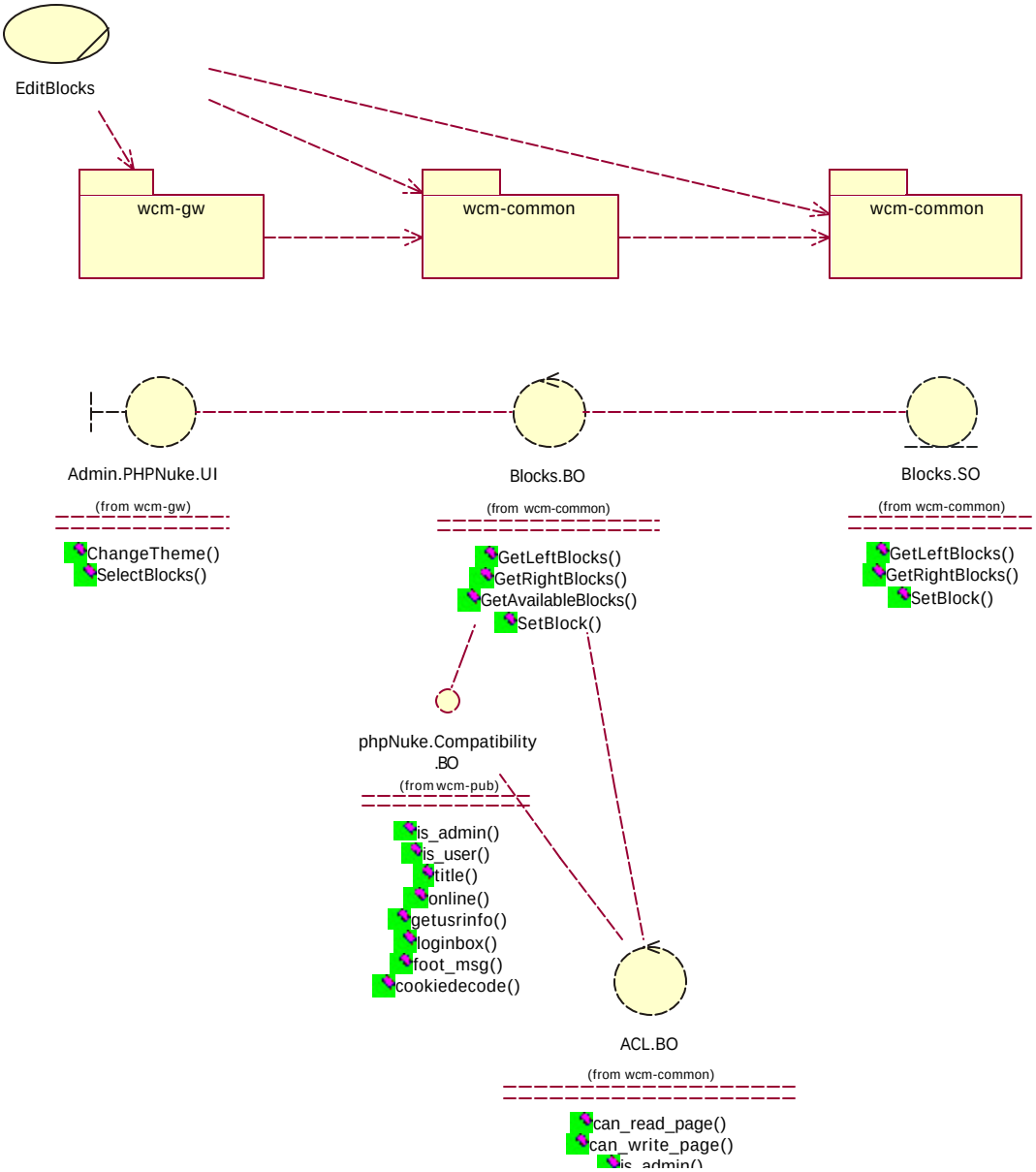
Contents:

Diagrams:


: Site Administrator
: Admin.PHPNuke.UI
: Blocks.BO
: ACL.BO
: Blocks.SO
Click Select Blocks link
1: SelectBlocks( )
2: is_admin( )
3: GetAvailableBlocks( )
4: GetLeftBlocks( )
5: GetLeftBlocks( )
6: GetRightBlocks( )
7: GetRightBlocks( )
Submit new block selections
8: SetBlock(string, Integer, Integer)
9: SetBlock(Array)


: Site Administrator
: Admin.PHPNuke.UI
: Error.UI
: Blocks.BO
: ACL.BO
: Blocks.SO
Click Select Blocks link
1: SelectBlocks( )
2: is_admin( )
3: GetAvailableBlocks( )
No available blocks
4: DisplayError( )


: Site Administrator
: Admin.PHPNuke.UI
: Error.UI
: Blocks.BO
: ACL.BO
: Blocks.SO
Click Select Blocks link
1: SelectBlocks( )
2: is_admin( )
Not an admin
3: DisplayError( )



ManagePage Use Case

Contents:

1) ManagePage Use Case Realization
   a) Sequence Diagrams
      i) Basic Course
      ii) Alternate Course 1
      iii) Alternate Course 2

b) Classes

i) Boundary

(1) Contributor.ManagePage.UI

ii) Control

(1) Categories.BO

(2) Pages.BO

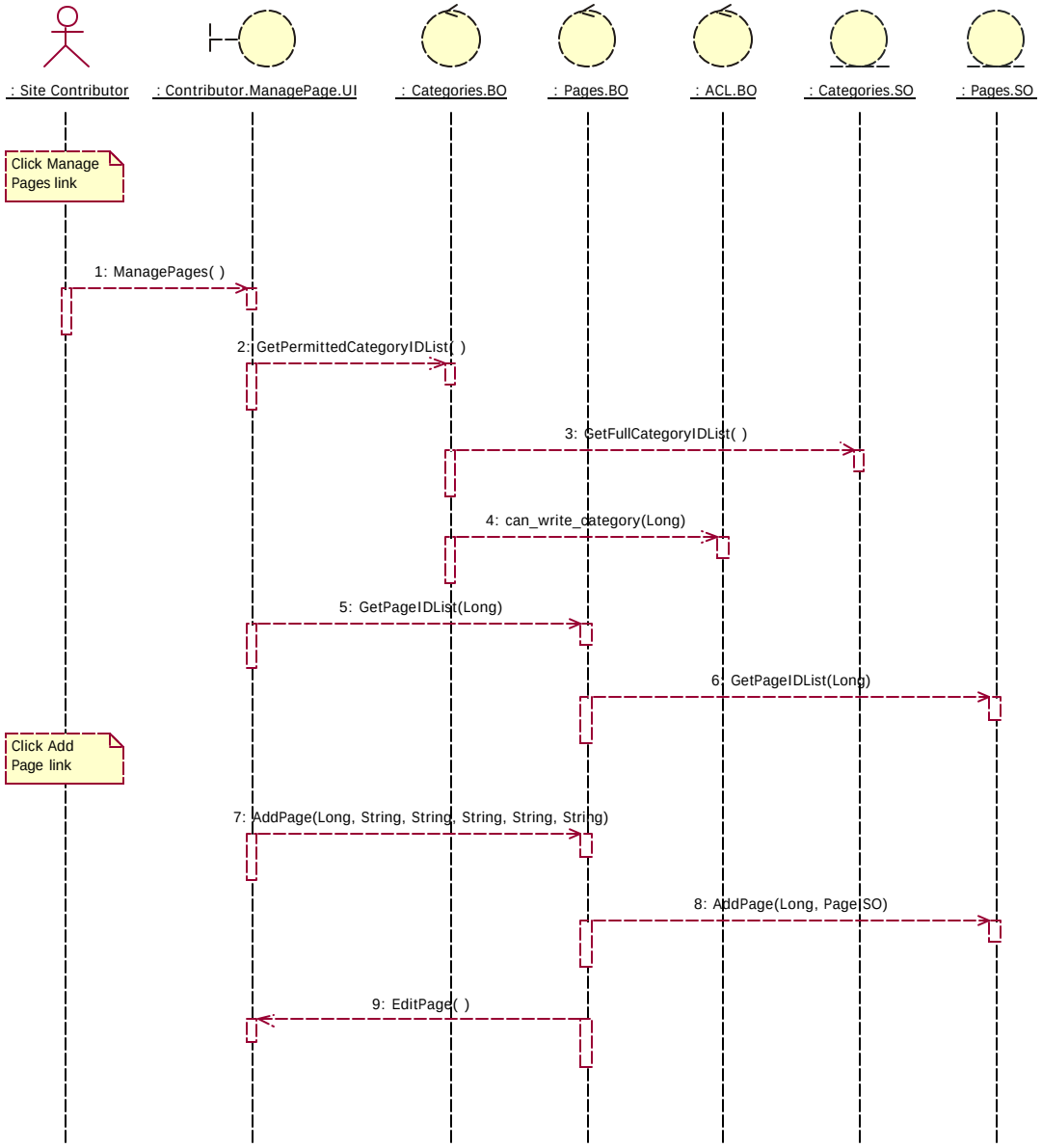
(3) ACL.BO

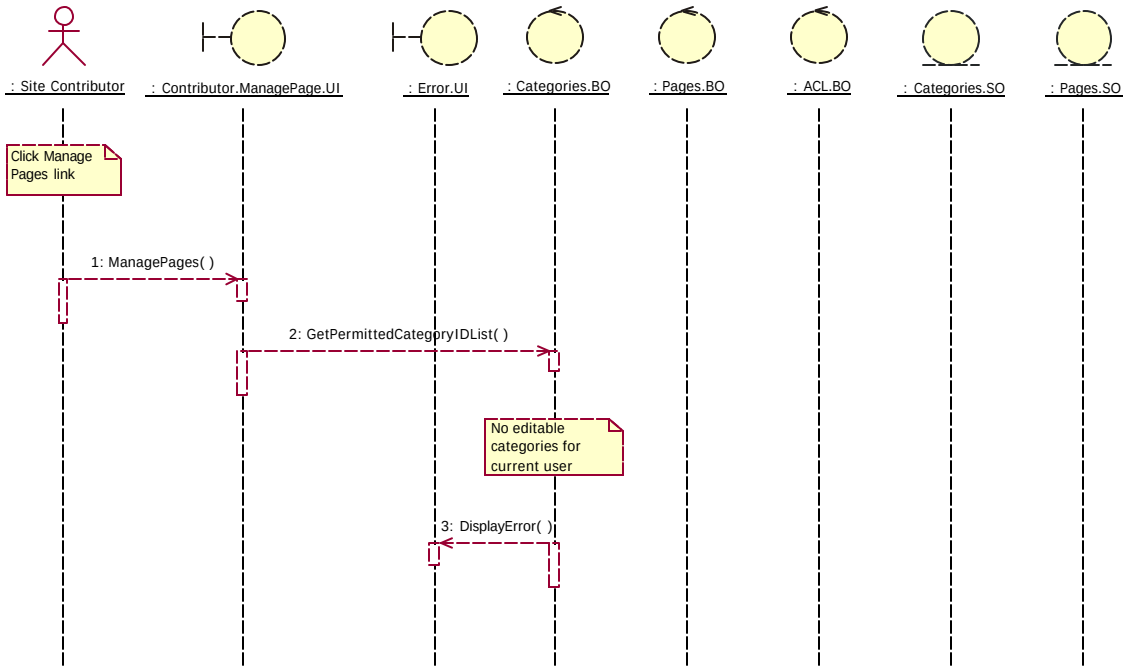
iii) Entity

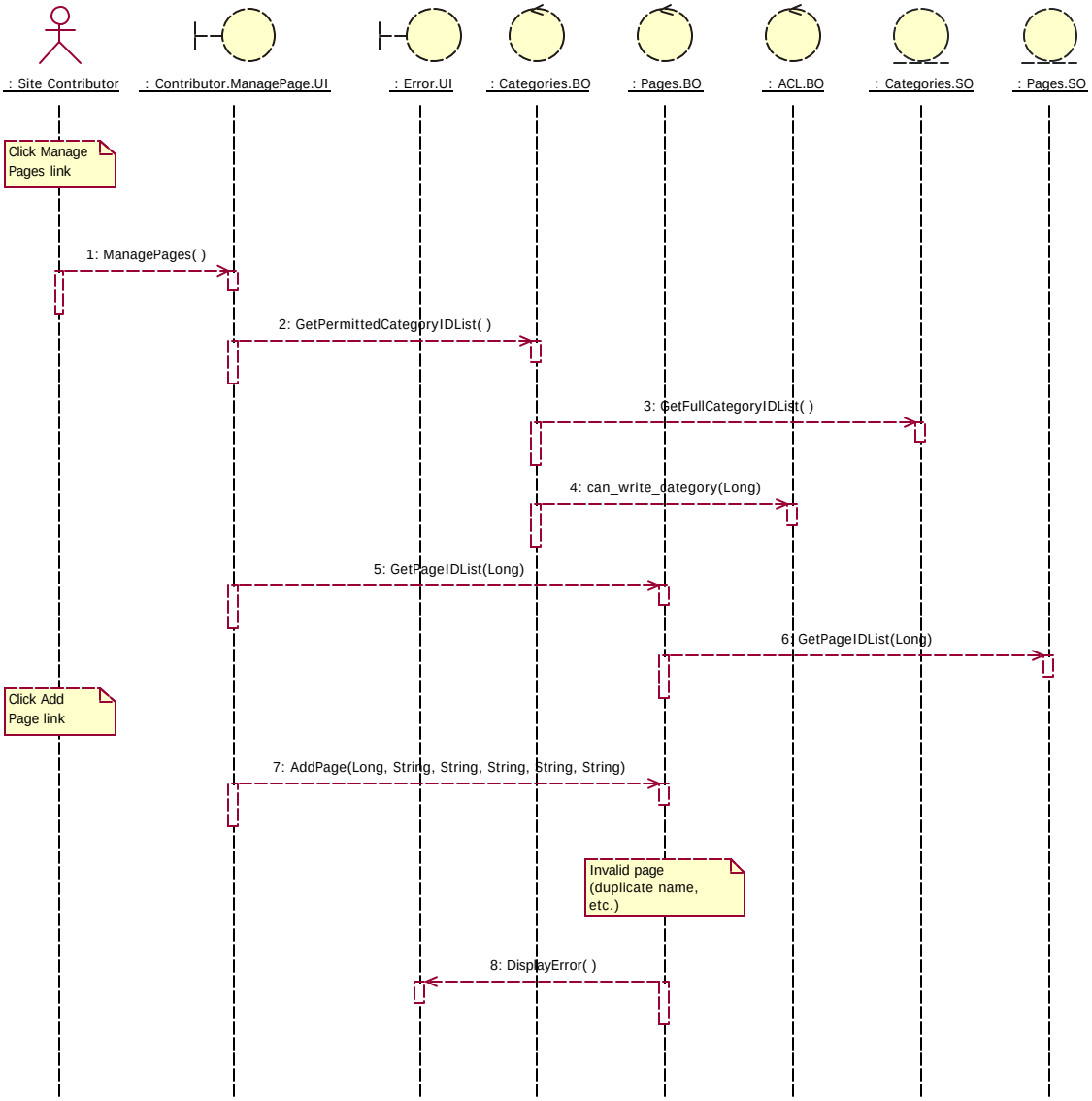
(1) Categories.SO

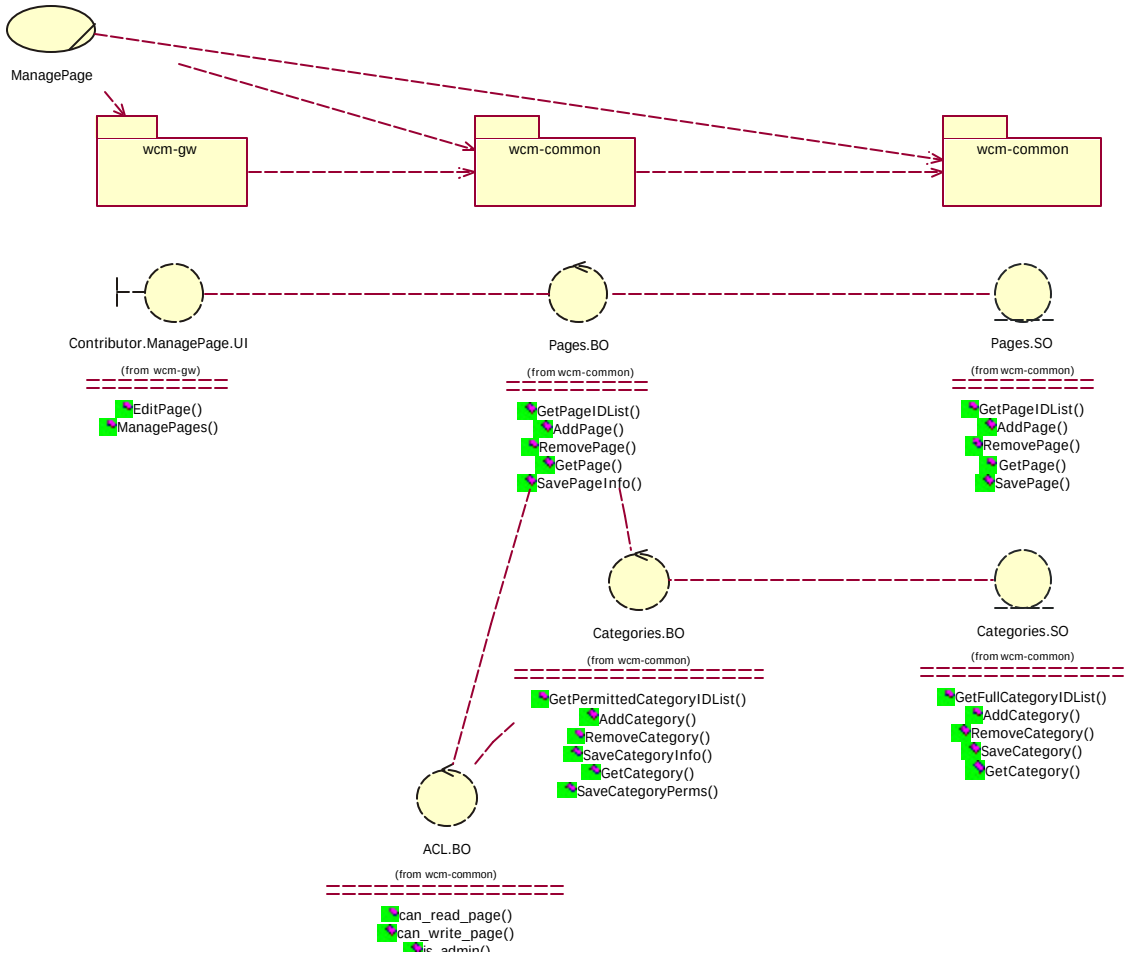
(2) Pages.So

Diagrams:


: Site Contributor
: Contributor.ManagePage.UI
: Categories.BO
: Pages.BO
: ACL.BO
: Categories.SO
: Pages.SO
Click Manage Pages link
1: ManagePages( )
2: GetPermittedCategoryIDList( )
3: GetFullCategoryIDList( )
4: can_write_category(Long)
5: GetPageIDList(Long)
6: GetPageIDList(Long)
Click Add Page link
7: AddPage(Long, String, String, String, String, String)
8: AddPage(Long, PageSO)
9: EditPage( )


: Site Contributor
: Contributor.ManagePage.UI
: Error.UI
: Categories.BO
: Pages.BO
: ACL.BO
: Categories.SO
: Pages.SO
Click Manage Pages link
1: ManagePages( )
2: GetPermittedCategoryIDList( )
No editable categories for current user
3: DisplayError( )


: Site Contributor
: Contributor.ManagePage.UI
: Error.UI
: Categories.BO
: Pages.BO
: ACL.BO
: Categories.SO
: Pages.SO
Click Manage Pages link
1: ManagePages( )
2: GetPermittedCategoryIDList( )
3: GetFullCategoryIDList( )
4: can_write_category(Long)
5: GetPageIDList(Long)
6: GetPageIDList(Long)
Click Add Page link
7: AddPage(Long, String, String, String, String, String)
Invalid page (duplicate name, etc.)
8: DisplayError( )



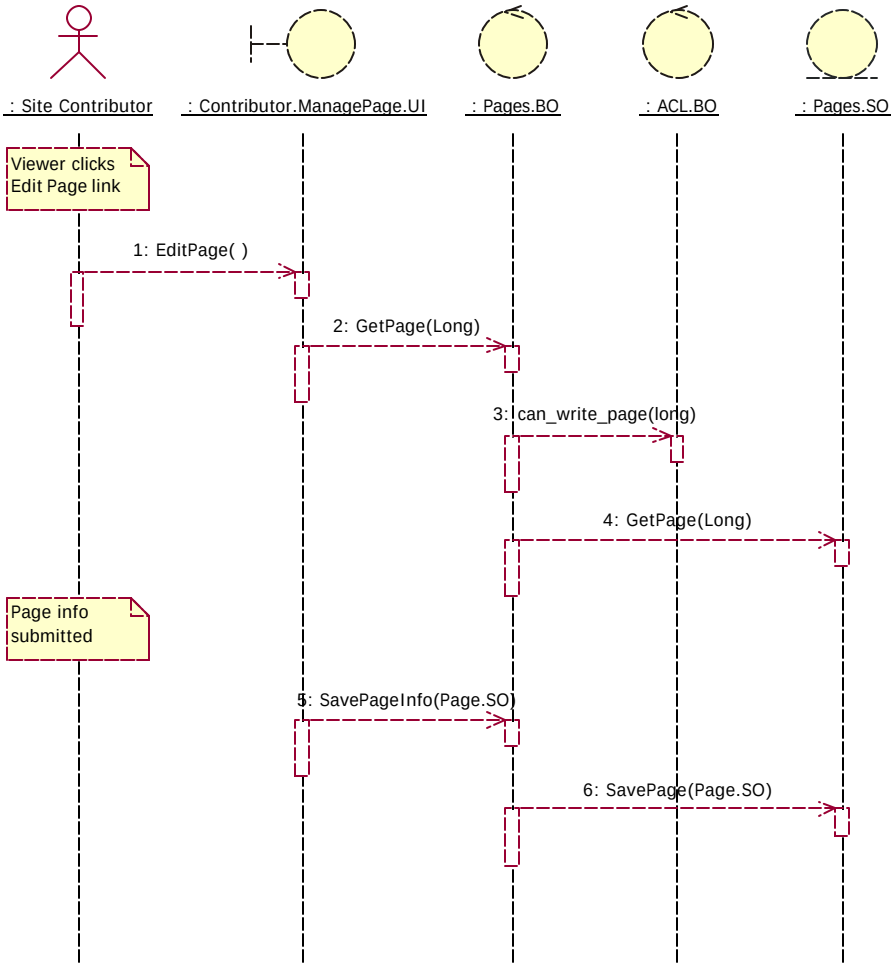
EditPage use Case

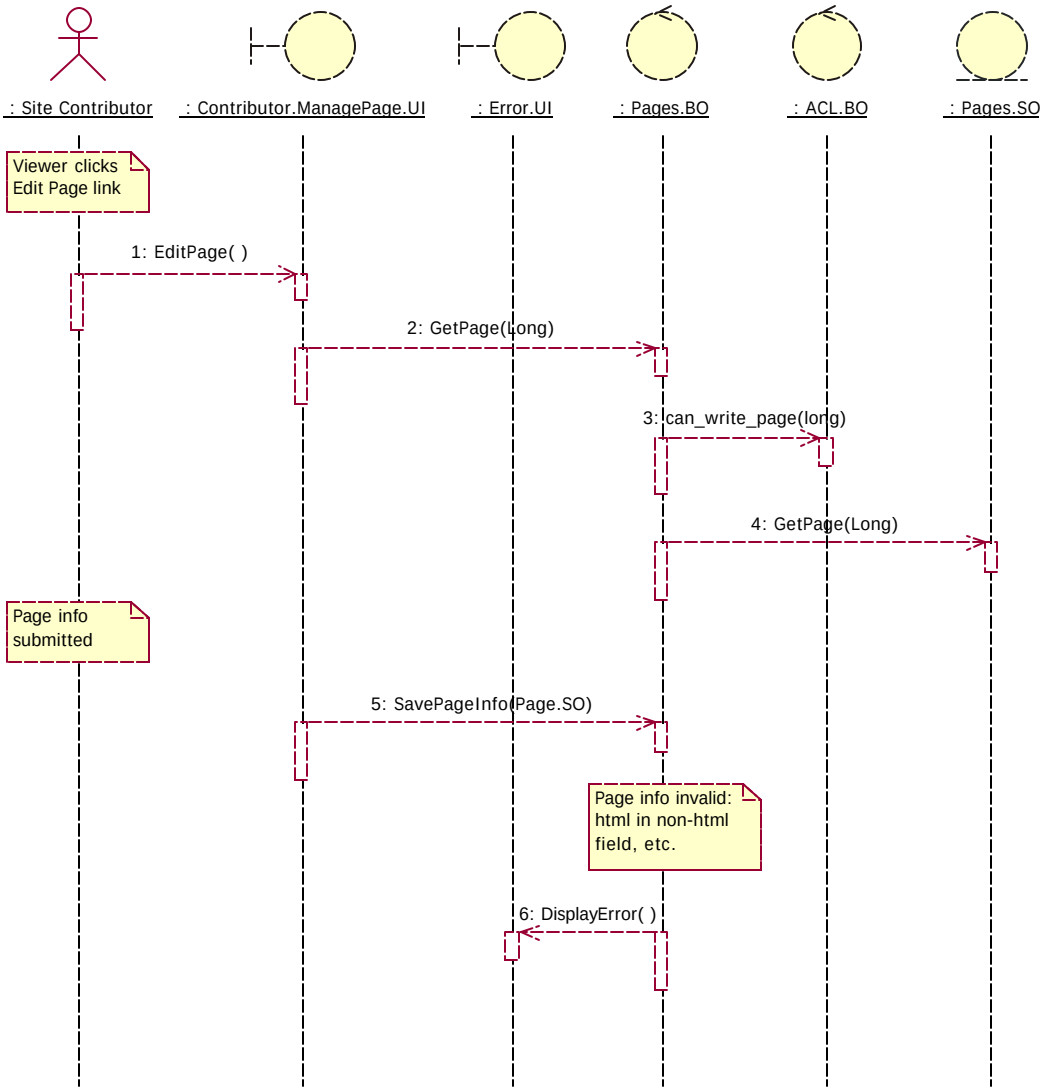
Contents:

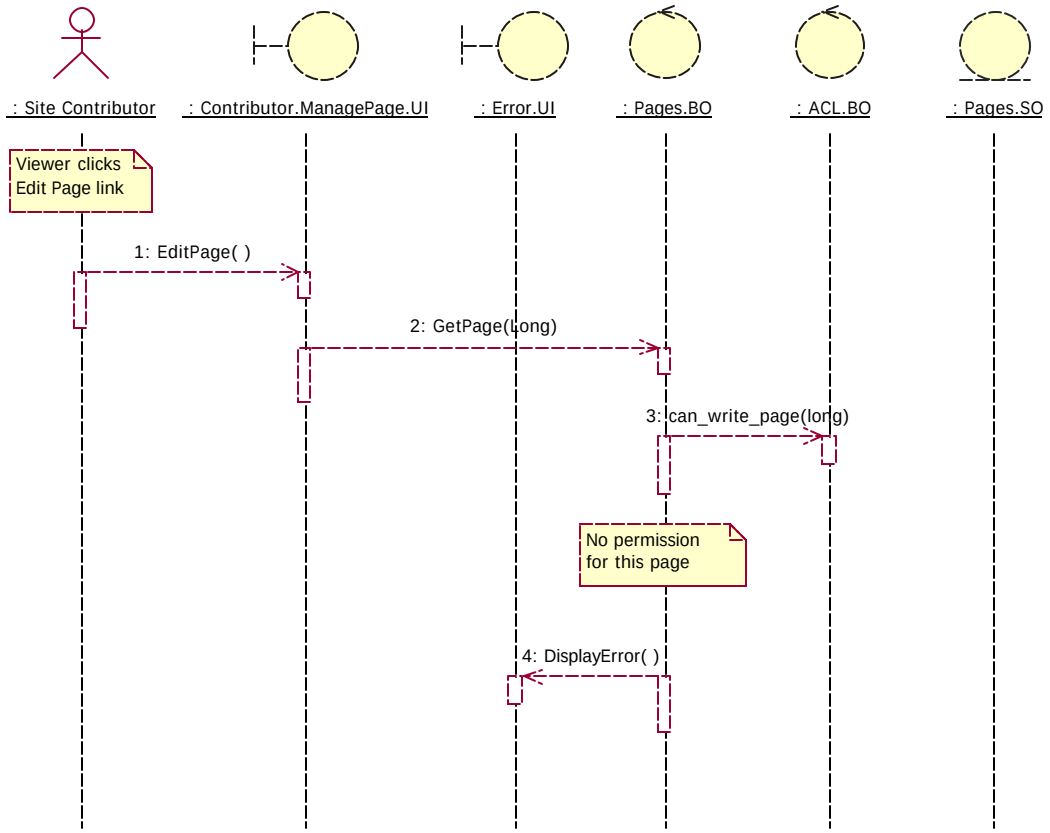
iii) Entity

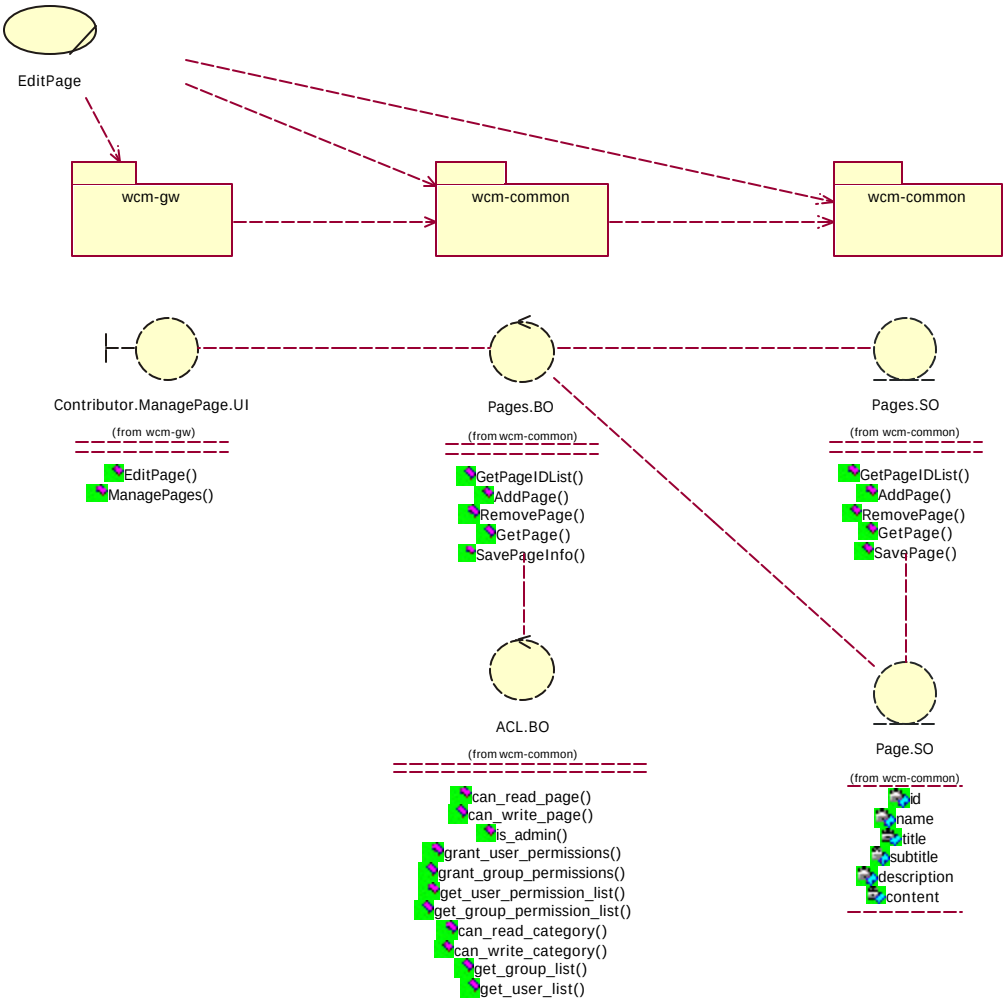
(1) Pages.So

Diagrams:


: Site Contributor
: Contributor.ManagePage.UI
: Pages.BO
: ACL.BO
: Pages.SO
Viewer clicks Edit Page link
1: EditPage( )
2: GetPage(Long)
3: can_write_page(long)
4: GetPage(Long)
Page info submitted
5: SavePageInfo(Page.SO)
6: SavePage(Page.SO)


: Site Contributor
: Contributor.ManagePage.UI
: Error.UI
: Pages.BO
: ACL.BO
: Pages.SO
Viewer clicks Edit Page link
1: EditPage( )
2: GetPage(Long)
3: can_write_page(long)
4: GetPage(Long)
Page info submitted
5: SavePageInfo(Page.SO)
Page info invalid: html in non-html field, etc.
6: DisplayError( )


: Site Contributor
: Contributor.ManagePage.UI
: Error.UI
: Pages.BO
: ACL.BO
: Pages.SO
Viewer clicks
Edit Page link
1: EditPage( )
2: GetPage(Long)
3: can_write_page(long)
No permission
for this page
4: DisplayError( )



GeneratePage use Case

Contents:

(1) PageGenerator.BO

(2) Theme.BO

(3) HeaderFooter.BO

(4) Blocks.BO

(5) Pages.BO

(6) ACL.BO

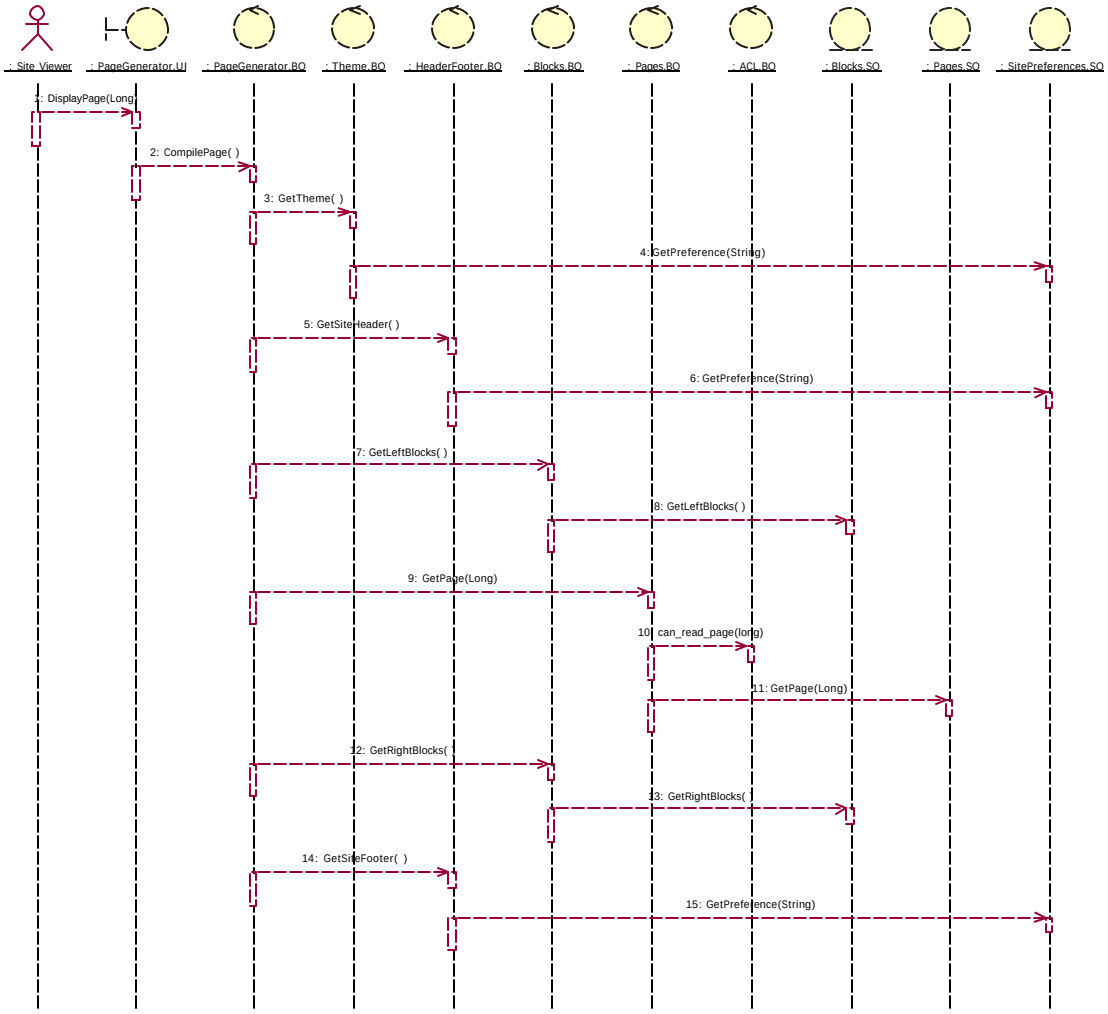
iii) Entity

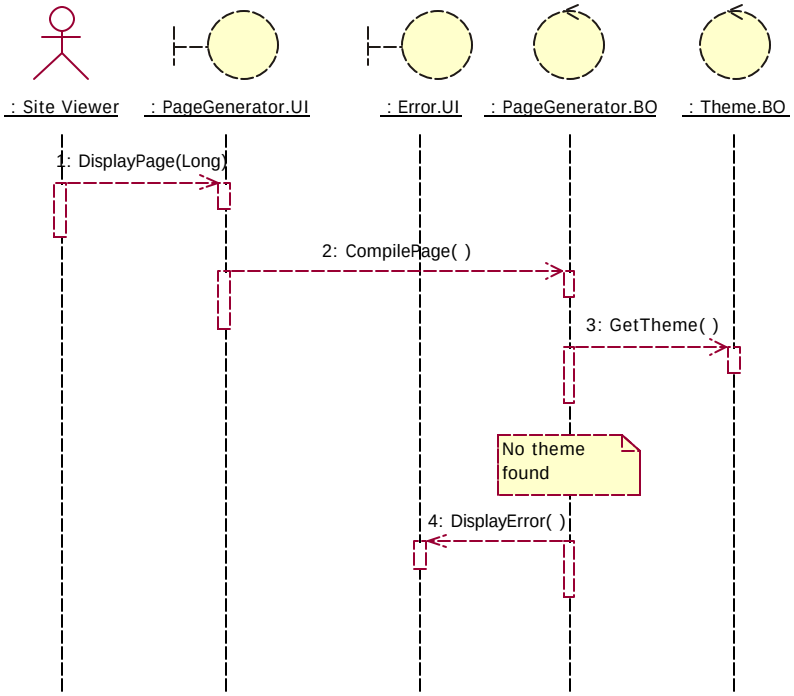
(1) Blocks.SO

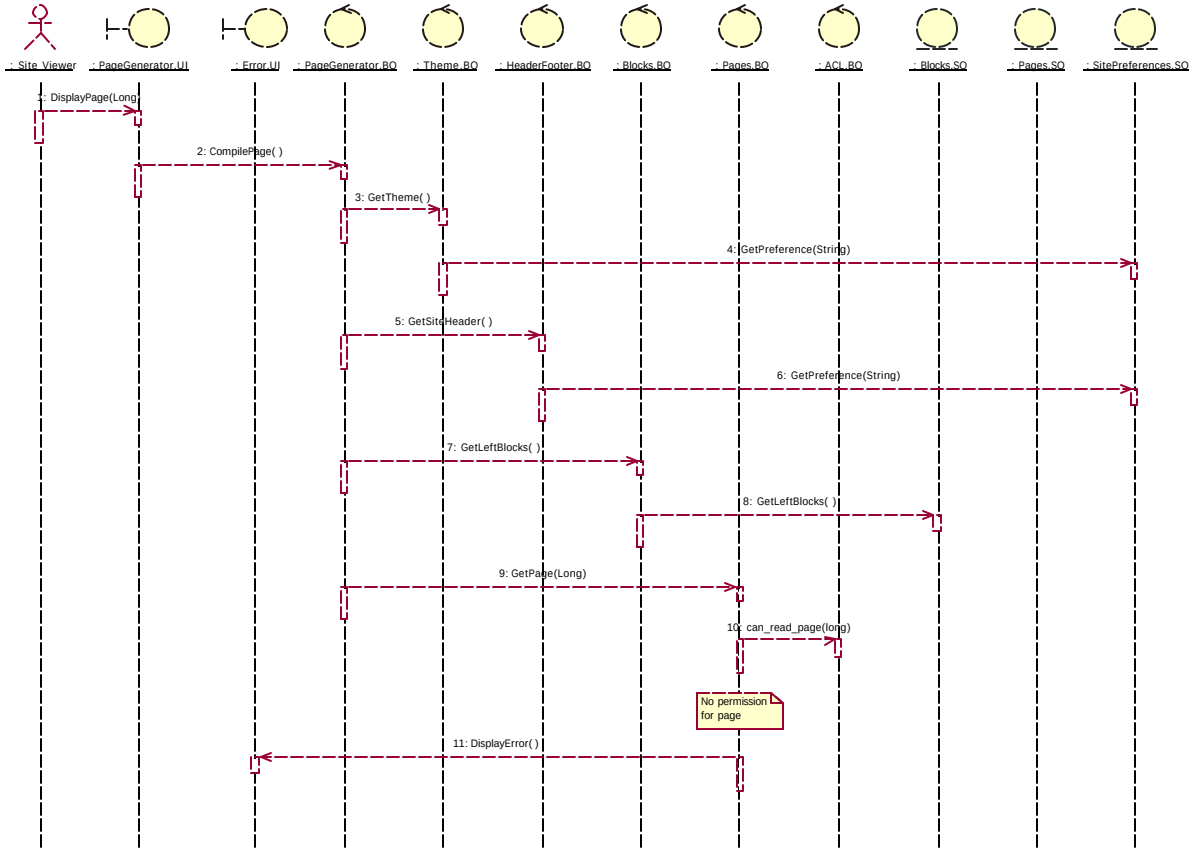
(2) Pages.SO

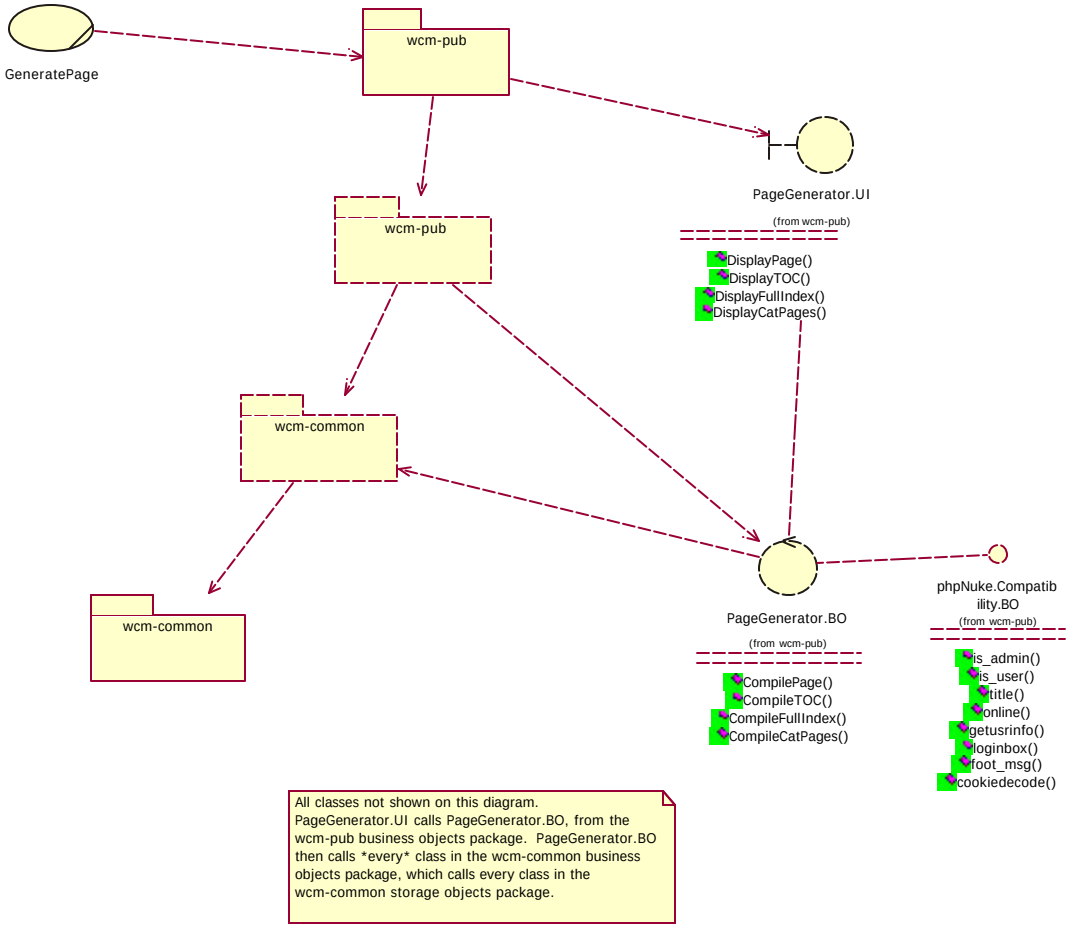
(3) SitePreferences.SO

Diagrams:


: Site Viewer
: PageGenerator.UI
: PageGenerator.BO
: Theme.BO
: HeaderFooter.BO
: Blocks.BO
: Pages.BO
: ACL.BO
: Blocks.SO
: Pages.SO
: SitePreferences.SO
1: DisplayPage(Long)
2: CompilePage( )
3: GetTheme( )
4: GetPreference(String)
5: GetSiteHeader( )
6: GetPreference(String)
7: GetLeftBlocks( )
8: GetLeftBlocks( )
9: GetPage(Long)
10: can_read_page(long)
11: GetPage(Long)
12: GetRightBlocks( )
13: GetRightBlocks( )
14: GetSiteFooter( )
15: GetPreference(String)

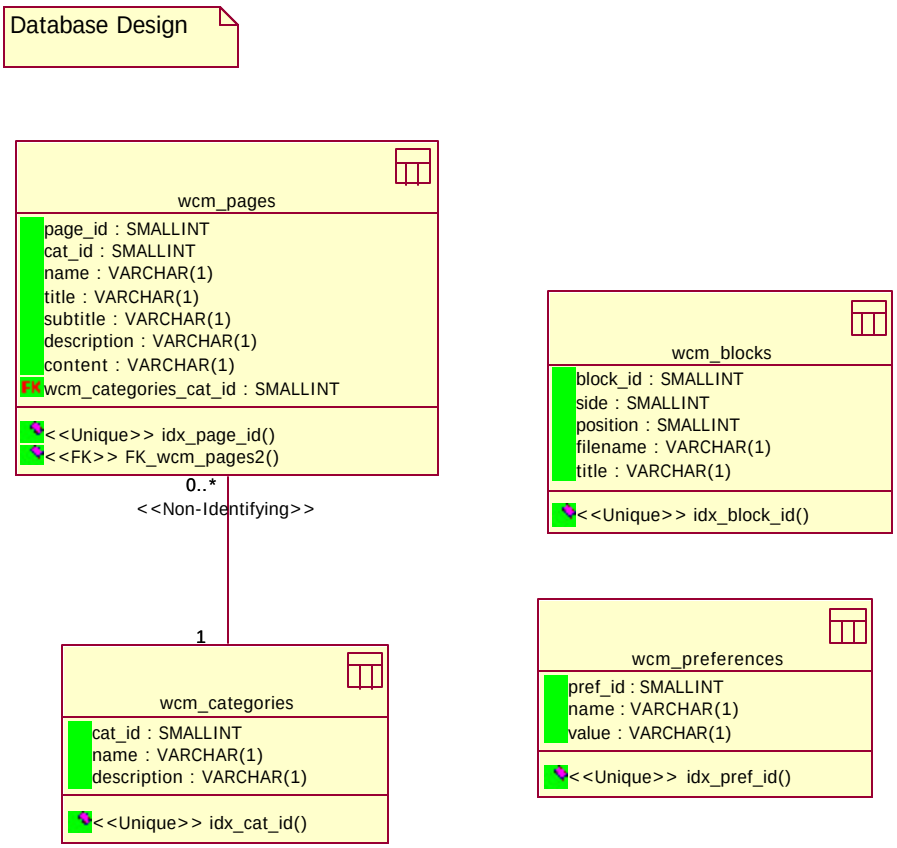

: Site Viewer
: PageGenerator.UI
: Error.UI
: PageGenerator.BO
: Theme.BO
1: DisplayPage(Long)
2: CompilePage( )
3: GetTheme( )
No theme found
4: DisplayError( )


: Site Viewer
: PageGenerator.UI
: Error.UI
: PageGenerator.BO
: Theme.BO
: HeaderFooter.BO
: Blocks.BO
: Pages.BO
: ACL.BO
: Blocks.SO
: Pages.SO
: SitePreferences.SO
1: DisplayPage(Long)
2: CompilePage( )
3: GetTheme( )
4: GetPreference(String)
5: GetSiteHeader( )
6: GetPreference(String)
7: GetLeftBlocks( )
8: GetLeftBlocks( )
9: GetPage(Long)
10: can_read_page(long)
No permission for page
11: DisplayError( )


GeneratePage
wcm-pub
PageGenerator.UI
(from wcm-pub)
DisplayPage()
DisplayTOC()
DisplayFullIndex()
DisplayCatPages()
wcm-pub
wcm-common
wcm-common
PageGenerator.BO
(from wcm-pub)
CompilePage()
CompileTOC()
CompileFullIndex()
CompileCatPages()
phpNuke.Compatibility.BO
(from wcm-pub)
is_admin()
is_user()
title()
online()
getusrinfo()
loginbox()
foot_msg()
cookiedecode()
All classes not shown on this diagram.
PageGenerator.UI calls PageGenerator.BO, from the wcm-pub business objects package. PageGenerator.BO then calls *every* class in the wcm-common business objects package, which calls every class in the wcm-common storage objects package.

SECTION 4: MISCELLANEOUS SPECS

Database Design



SECTION 5: TEST CASE PLAN

Unit Testing

Test Case ID	Items being Tested	Input(s)	Expected Outputs	Actual Outputs
1	ManagePages() in Contributor_ManagePage_UI	None	Category Managing Page should be generated.	Correct!
2	EditPage() in Contributor_ManagePage_UI	Category ID and Page ID	Page Editor page should be generated	Correct!
3	GetPermittedCategoryIDList() in Categories.BO	None	An array of category ids	Correct!
4	can_read_page($page_id) in ACL_BO	Page ID	True, if the contributor has permission to read the page. Else, return false	Correct!
5	can_write_page($page_id) in ACL.BO	Page ID	True, if the contributor has the permission to write	Removed from implementation
6	GetFullCategoryIDList() in Categories.SO	None	An array of category ids	Correct!
7	GetPageIDList($cat_id) in Pages.BO	Category ID	Array of page IDs	Correct!
8	AddPage($cat_id)in Pages.BO	Category ID	Return new page id, which is created in DB.	Correct!
9	RemovePage($cat_id) in Pages.BO	Category ID	True, if the page is removed, else False	Correct!
10	GetPage($page_id) in Pages.BO	Page ID	A Page class, which the page id.	Correct!
11	SavePageInfo($page) in Pages.BO	Page class	True, if the page class is saved in the data backend. Else, False	Correct!
12	GetPageIDList($cat_id) in	Category ID	Array of available pages	Correct!

	Pages.SO		under the category id.	
13	AddPage($cat_id) in Pages.SO	Category ID	New page id created in DB	Correct!
14	RemovePage($page_id) in Pages.SO	Page ID	The query call from the DB	Correct!
15	GetPage($page_id) in Pages.SO	Page ID	If the page id exists, a page class with the page ID. Else, return False.	Correct!
16	SavePage($page) in Pages.SO	Page class	True, if the page class is saved.	Correct!
17	ChangeTheme() in Admin.PHPNuke.UI	None	Theme selection page should appear.	Implemented in next version
18	SelectBlocks() in Admin.PHPNuke.UI	None	Blocks managing page should appear.	Implemented in next version
19	is_admin() in ACL.BO	None	Allow the user to go to the request page.	Correct!
20	GetLeftBlocks() in Blocks.Bo	None	The left side blocks should be displayed in the block selection page.	Implemented in next version
21	GetRightBlocks() in Blocks.Bo	None	The right side blocks should be displayed in the block selection page.	Implemented in next version
22	GetAvailableBlocks() in Blocks.BO	None	Blocks information should be retrieved from the database. Either blocks are available or blocks can't be retrieved.	Implemented in next version
23	SetBlock($filename, $side, $position): Boolean in Blocks.BO	Block filename, Side (left/right), Position on side (1-X)	Blocks should be displayed in the way as the administrator assigns them.	Implemented in next version
24	GetSiteHeader() in HeaderFooter.BO	Variable call.	The current header of the site should be displayed in	Correct!, displays "Team

			the text field.	X project."
25	GetSiteFooter() in HeaderFooter.BO	Variable call.	The current footer of the site should be displayed in the text field.	Correct!, displays "Copyrighted 2002"
26	SetSiteHeader() in HeaderFooter.BO	Header (Team X project)	The Administrator's input in the header text field should be displayed in the final generated page.	Correct!, displays "Team X project."
27	SetSiteFooter() in HeaderFooter.BO	Footer (Copyrighted 2002)	The Administrator's input in the footer text field should be displayed in the final generated page.	Correct!, displays "Copyrighted 2002"
28	SetTheme() in Theme.BO	Information of Theme	The site should be looked like the theme the administrator selected.	Implemented in next version
29	GetTheme() in Theme.BO.	Information of Theme	The Theme selected by the administrator should be retrieved.	Implemented in next version
30	GetAvailableTheme() in Theme.BO	None	The themes in the database should be retrieved or a error message should be displayed	Implemented in next version
31	GetLeftBlocks() in Blocks.SO	None	The chosen block should be retrieved from the database.	Implemented in next version
32	GetRightBlocks() in Blocks.SO	None	The chosen block should be retrieved from the database.	Implemented in next version
33	SetBlock() in Blocks.SO	None	The block selection setting set by the administrator should be saved into the database.	Implemented in next version
34	SetPreference($name, $value:	name and	The preference set by the	Correct!

	String) SitePreference.SO	value	administrator should be saved into the database.	
35	GetPreference($name) SitePreference.SO	Name	The requested preference should be retrieved from the database then returned to the admin	Correct!

Integration Testing

Test Case ID	Items being Tested	Input(s)	Expected Output(s)	Actual Output(s)
1	Add new category	1. Go to the "Manage Categories" page in the administrative section of the website. 2. Select "Add new Category" link 3. Fill in all the fields with "valid" inputs. 4. Click "Save" button.	1. The newly added category should be available (either read and/or write) to all the groups/individuals who were given those access permissions. 2. The category should NOT be available (either read and/or write) to groups/individuals who are not given those access permissions. 3. The name of the category should be listed in the Administrative Category Management page so that the administrator can edit the categories settings and/or permissions.	1. Correct! 2. Correct! 3. Correct!
2	Edit Category	1. Go to the "Manage Categories" page in the administrative section of the website. 2. Select "Edit Category" link	1. The edited category should be available (either read and/or write) to all the groups/individuals who were given those	1. Correct! 2. Correct! 3. Correct!

		3. Edit all fields w/ correct inputs 4. Click “Save” button.	access permissions. 2. The category should NOT be available (either read and/or write) to groups/individuals who are not given those access permissions. 3. The new name and description of the category should be listed in the Administrative Category Management page so that the administrator can edit the categories settings and/or permissions.	
3	Delete Category	1. Go to the “Manage Category” page in the administrative section of the website. 2. Select “Delete Category” for a category. 3. An alert box should pop up; hit “ok” to delete the category.	1. The category should be removed from the phpGroupWare database. 2. All pages contributed to the category should also be removed. 3. The category should no longer be listed on the contributor page, nor should it be listed in the Administrative Category Management Page.	1. Correct! 2. Correct! 3. Correct!
4	Header / Footer Change	1. Request for headerfooter page.	Header: Team X project. Footer: Copy righted 2002	Header: Team X

		2. Do a permission checks. 3. After permission is checked, headerfooter editing page appears. 4. Input “Team X project” into header text field and “Copy righted 2002” into footer text field. Then save it.		project. Footer: Copyrighted 2002
5	Select Blocks Page	1. Request for block selection. 2. Do a permission checks. When permission is checked, the selection page shows up, block list has been retrieved from the database. 3. The admin changes the selections.	The blocks in the final generated page should look like the one the admin positioned	Implemented in next version
6	Theme Selection page	1. Request for theme selection page. Permission checks. 2. When the permission is granted, the theme list would be loaded from the database. 3. The Admin can choose the desired theme for the site.	The theme of the site should be exactly the same as the one choose by the admin.	Implemented in next version
7	Add new page	1. Click add new page button in the category-managing page. 2. Provide the content of the page in the page managing page.	1. The new added page should be viewable to the contributor. 2. The name of the page should be included in the list of pages in the	1. Correct! 2. Correct!

		3. Click “Save” button.	category in category-managing page.	
8	remove a page	Clicking “Remove” button in category-managing page.	1. The removed page should be removed from phpGroupWare data backend. 2. The removed page name should be removed from the list of pages in the category.	1. Correct! 2. Correct!
9	Edit a page	1. Click “Edit” button next to the page name in category managing page. 2. Edit the content of the page. 3. Click “Save” button.	1. After the Edit button is clicked, the page editor page should be generated. 2. After clicking “Save” button, the edited page should be appeared in a separate window with confirmation.	1. Correct! 2. Correct!

System Testing

Test Case ID	Items being Tested	Input(s)	Expected Output(s)	Actual Output(s)
1	Contributor_ ManagePage_ UI	Go to the category managing page.	Category managing page should be generated with list of categories and the list of the pages.	Correct!
2	Contributor_ ManagePage_ UI	Go to the category managing page. Click “Add new page” or “Edit” button.	After clicking “Add new page” or “Edit” button, the page managing page should be viewable with proper content.	Correct!
3	add page functions in Pages_BO, Pages_SO & ACL_BO	Add number of pages.	The titles of the pages should be added to the category. The write permission of the contributor should be checked whenever a page is added.	Correct!
4	Editing page functions in Pages_BO, Pages_SO & ACL_BO	Edit a page	The write permission of the contributor should be check before the page managing page is viewable. The edited page should be changed to the edited information.	Correct!
5	Removing a page in Pages_BO, Pages_SO & ACL_BO	Remove a page	The write & read permission of the contributor should be checked whenever a page is removed from the category.	Correct!

			The page title should be removed from the category in the category-managing page. The page should not be viewable.	
6	Admin_ PHPNuke_UI	Select Blocks selection page.	The blocks list should be generated and the selection and position pull down menu should be displayed next to each block	Implemented in next version
7	Setblocks() in both Block_BO and Block_SO	Save blocks setting.	The Blocks setting should be saved into the database. And the displayed blocks should be displayed as the way they are positioned.	Implemented in next version
8	theme list database retrieval in both Theme_BO and Theme_SO	Retrieve the theme list from the database.	The Permission should be checked before the retrieving process. The list will then retrieve and generate for the admin to use;	Implemented in next version
9	blocks data retrieval in both Blocks_BO and Blocks_SO	Retrieve the block list from the database	Before the retrieving process, the permission of the user will be checked. The block list will then retrieve and allow the user to select them.	Implemented in next version
10	save theme function in both Theme_BO and Theme_SO	Select a theme and then save it.	The theme will be saved into the database. The display page should display the same theme as selected.	Implemented in next version

			When the saved theme is called, the saved theme should be retrieved and returned to the user's screen.	

SECTION 6: USER INTERFACE DIAGRAMS

Administrative UI Diagrams

Figure 1: Administrative Menu UI Diagram:

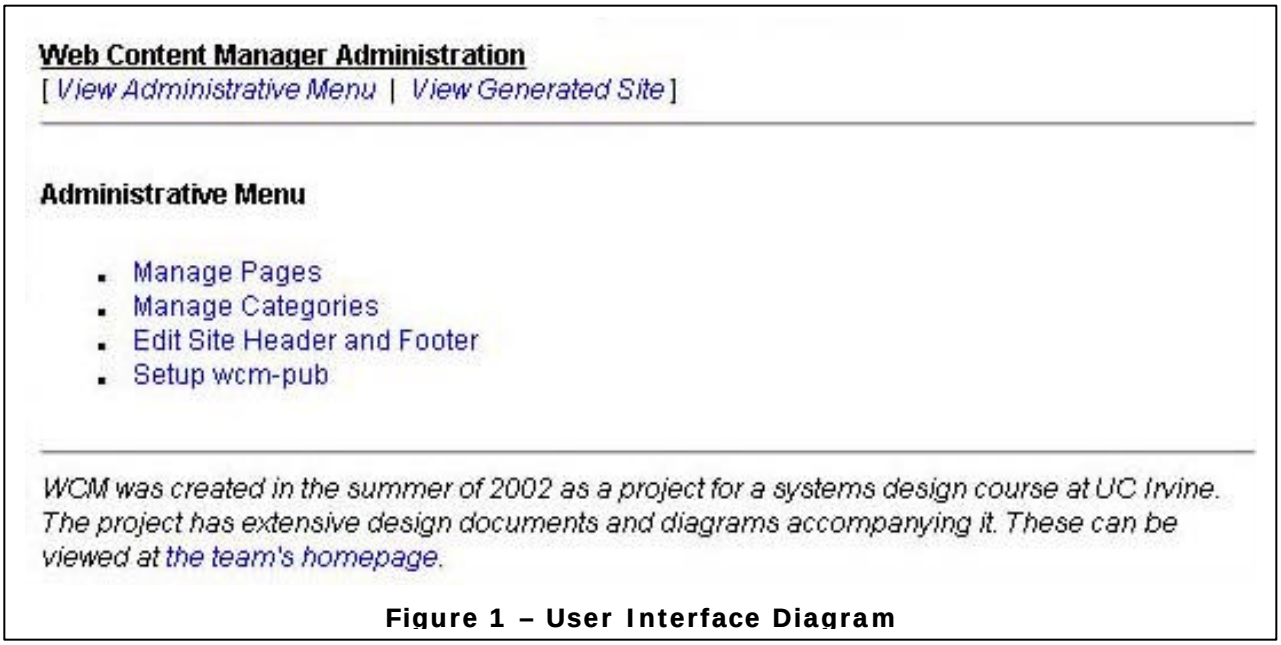


Figure 1 – User Interface Diagram

Figure 2: Category Manager UI Diagram 1:

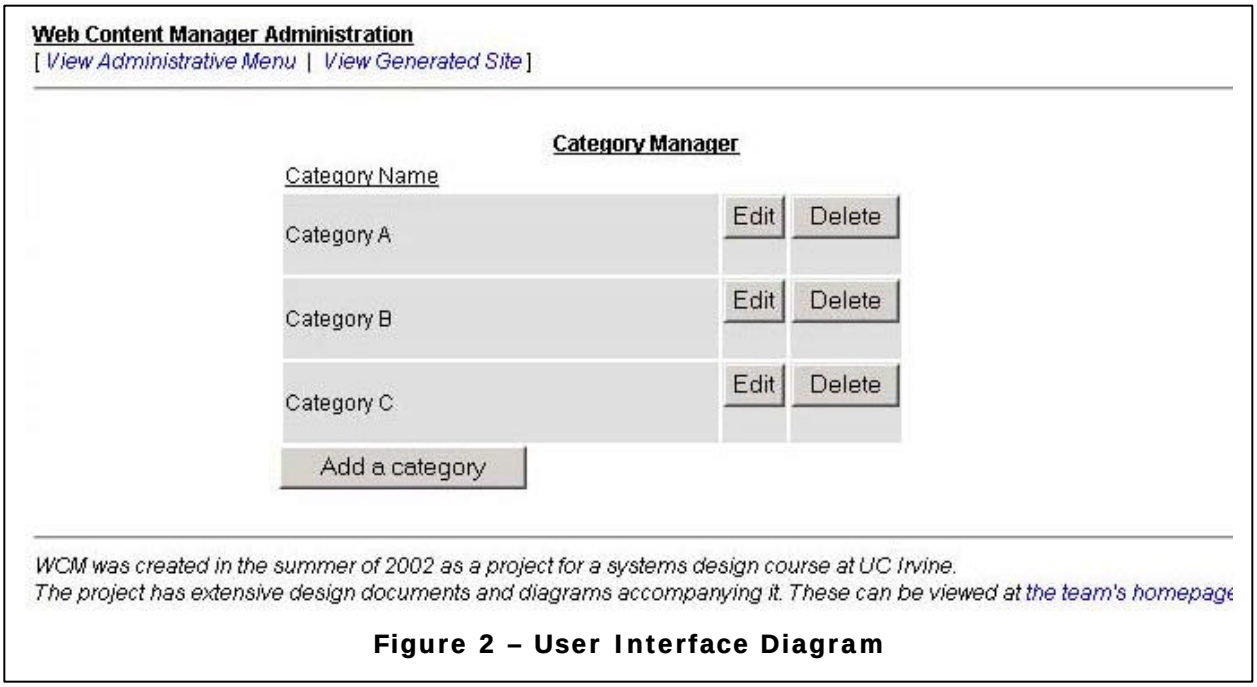


Figure 2 – User Interface Diagram

Figure 3: Add/Edit Category UI Diagram 2:

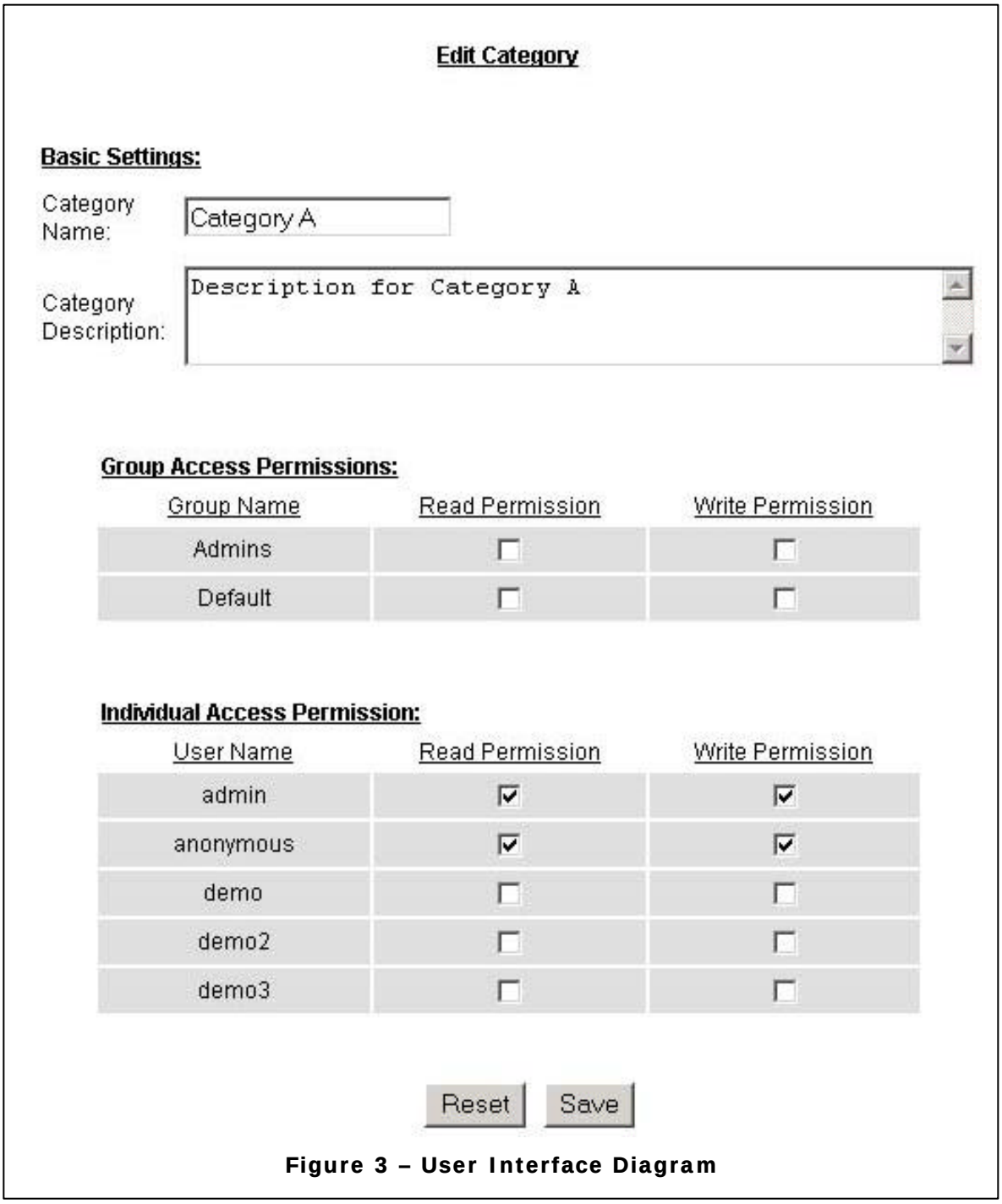


Figure 3 – User Interface Diagram

Figure 4: Edit Header / Footer Content UI Diagram:

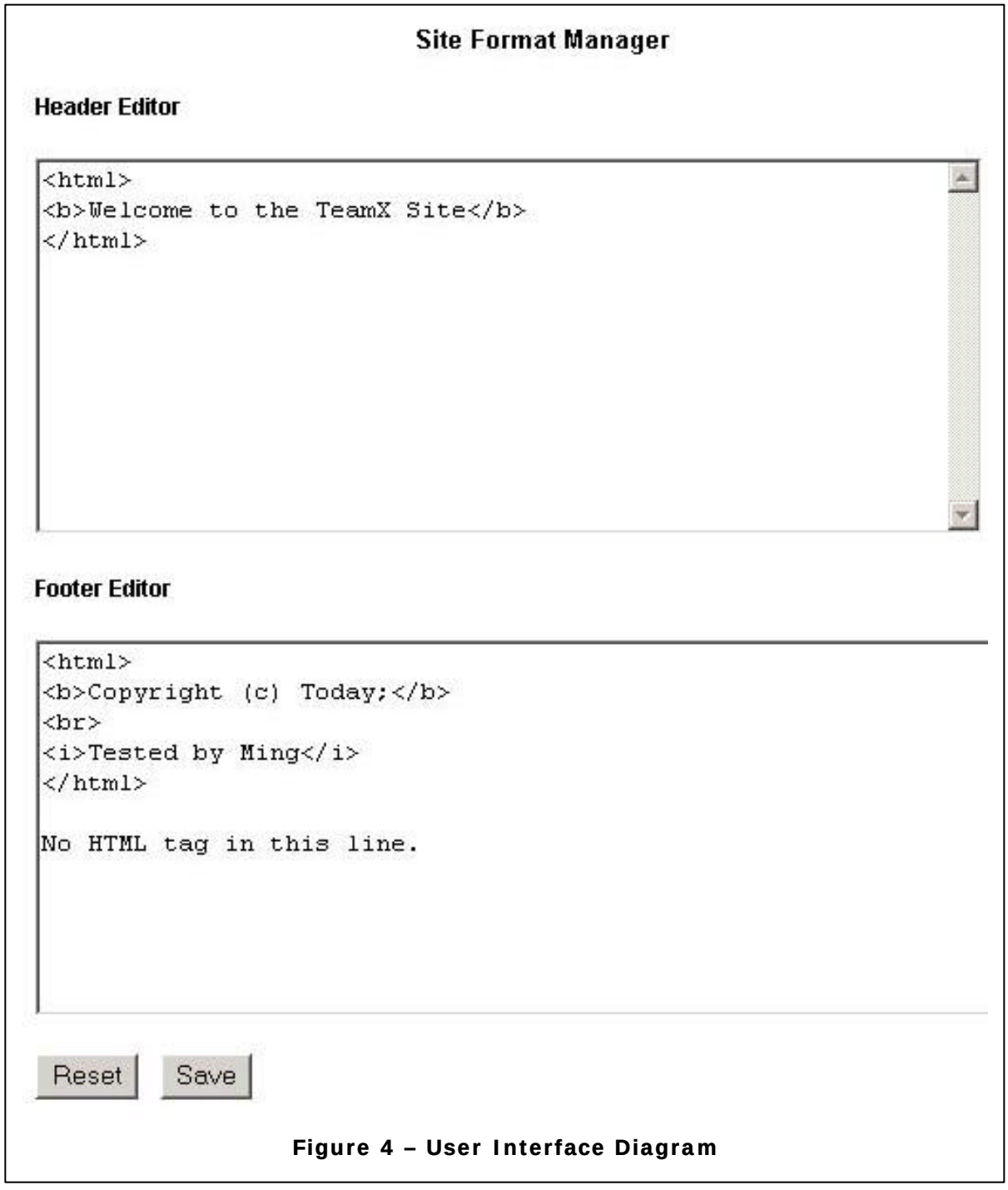


Figure 4 – User Interface Diagram

Contributor UI Diagrams

Figure 5: Page Manager UI Diagram:

Figure 5 – User Interface Diagram

Figure 6: Add/Edit Page UI Diagram:

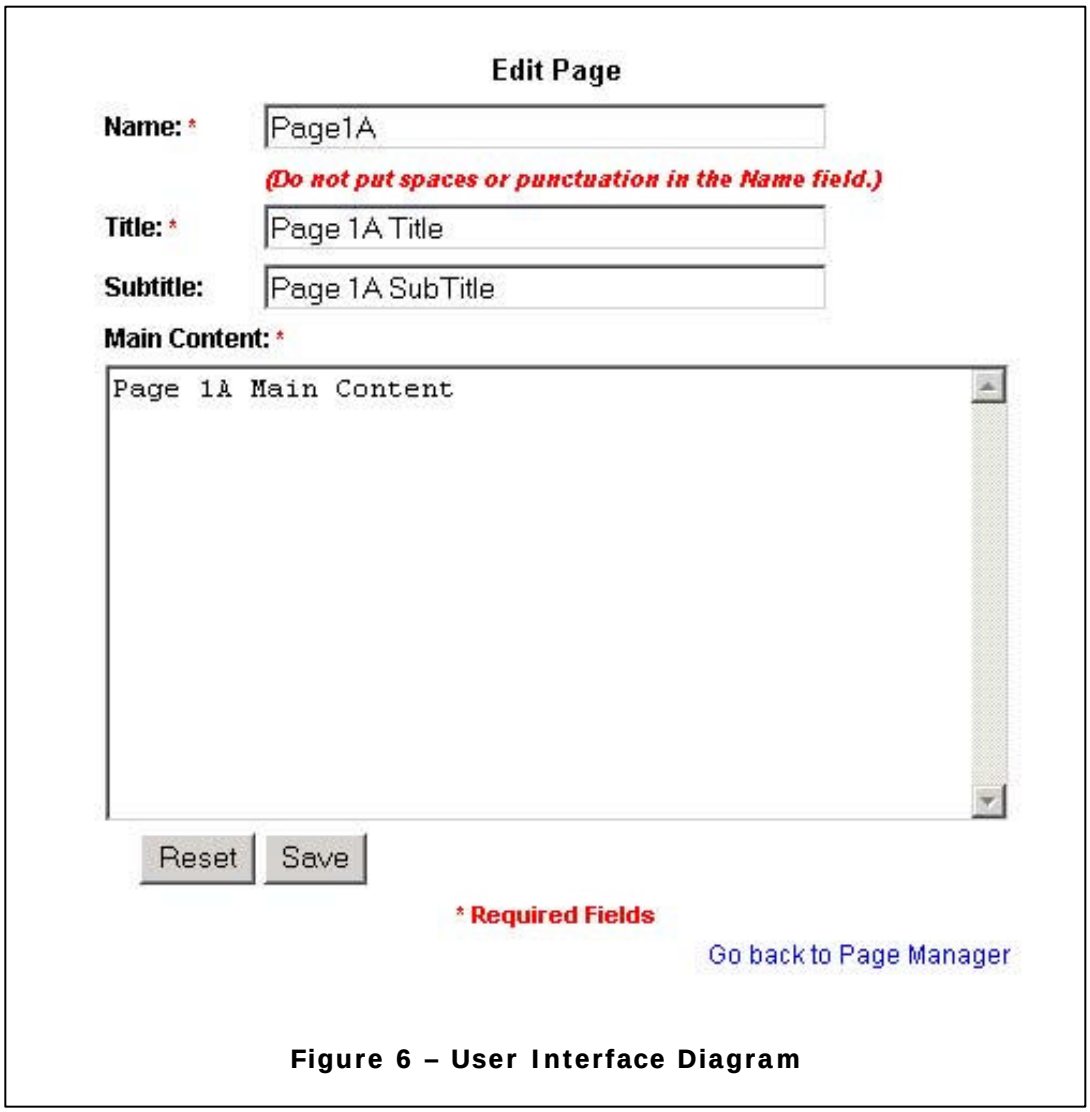


Figure 6 – User Interface Diagram

Page Generation UI Diagrams

Figure 7: Generated Page: Site Contents UI Diagram:

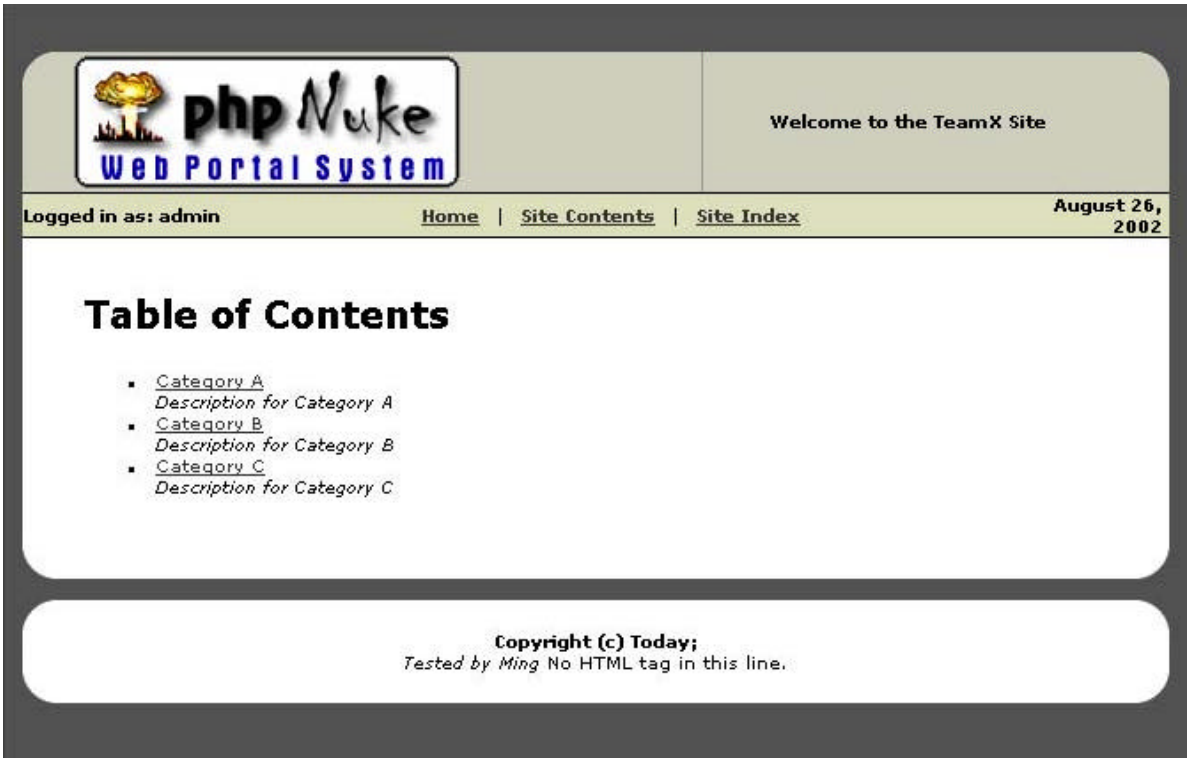


Figure 7 – User Interface Diagram

Figure 8: Generated Page: Site Index UI Diagram:

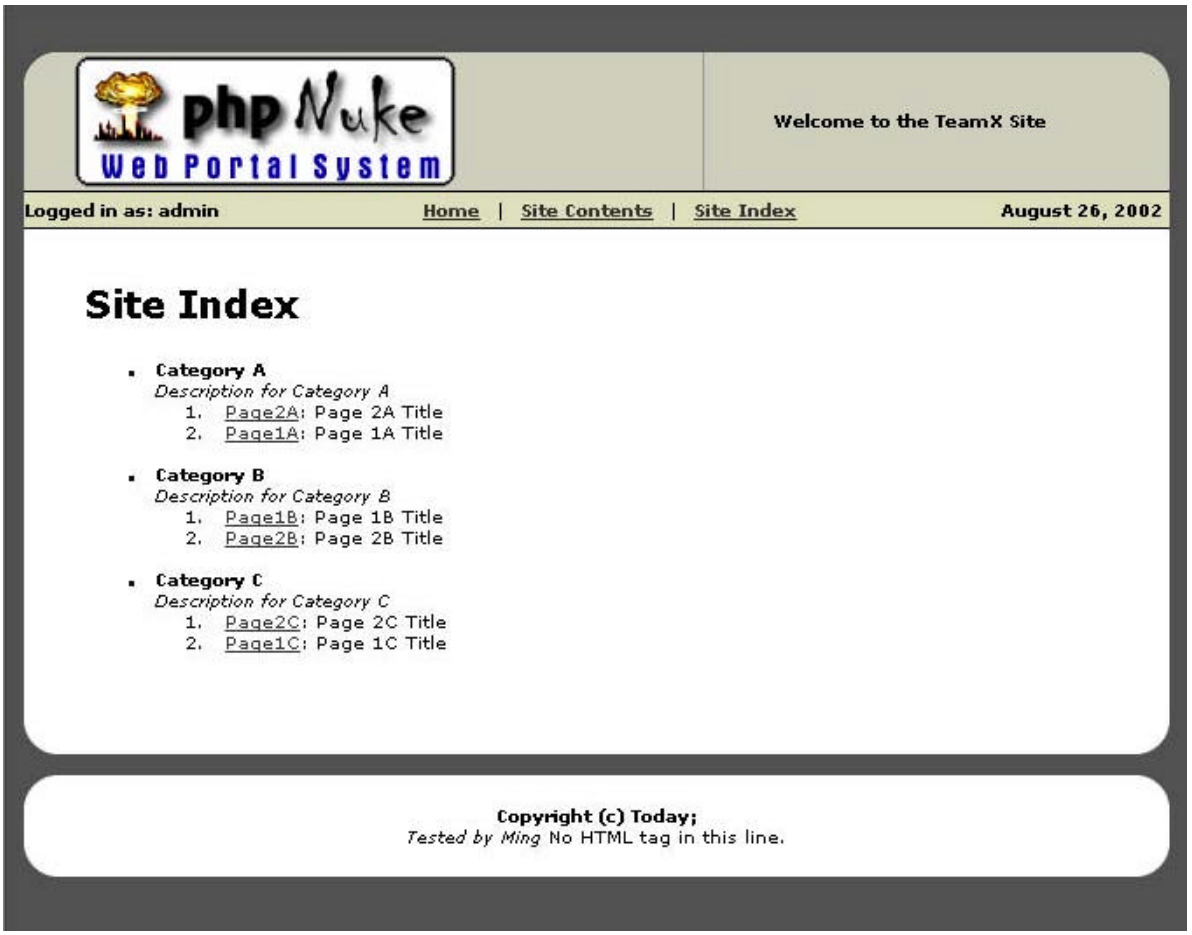


Figure 8 – User Interface Diagram

Figure 9: Generated Page: Category Table of Contents UI Diagram:

Figure 9 – User Interface Diagram

Figure 10: Generated Page UI Diagram:

Figure 10 – User Interface Diagram