TEAM 50 AGS

prev. 100 but changes needed!

SE USE CASE MODEL (Iteration)

Ø No changes 50 points

No work assigned

No SVN!

Based on RED TEAMS DELIVERABLES Feedback

RED Original # points

10 "Requirements" (11) pages (149) Line Numbers

or "Multiple Requirement" page (4) (48) Line Number (last)

– 20 points

30 Step 1 (prev. 4) Compilable URL X (4) – 10 points

Actors: 7

Customer Visitor Owner Salesperson Customer Service Rep dbi Artist

Step 2 (prev. 14) Compilable URL X (14) – 20 points

30 Step 3 USE CASE DIAGRAM MODEL (X) "Cut"s "Paste"s (One!) Not in Word (X) – 30 points

100% (X)

30 Step 4 USE CASE DESCRIPTION TEMPLATE X – 30 points

If not changed at ALL from feedback – 100 points or Ø

TEAM5OAGS

SE Team Project with Line Numbers
TEXTUAL ANALYSIS for Requirements Workflow
UML USE CASE DIAGRAM

# of Actors: 4

Table of Actors:

A1	Owner
A2	Salesperson
A3	Customer
A4	DBA
A5	Guest
A6	Customer Service Operator
A7	Worker

# of UCs: 14

Table of UCs:

UC1	request customer report
UC2	request Artist report
UC3	request customer artist preferences report
UC4	purchase new art
UC5	repurchase art from customer
UC6	request past purchase report
UC7	request past purchase artwork location report
UC8	request artist and works report
UC9	request speed of sale report
UC10	request current inventory report
UC11	modify account
UC12	create account
UC13	add new artist
UC14	add new customer

UC15	request online chat
UC16	request search
UC17	request art framing service
UC18	request work order report
UC19	log in
UC20	Sell art

Version 2.0

1

Team Project: TEAM5OAGS (OnLine Art Gallery System) Web Site

The **TEAM5OAGS** web site should be designed using these principles:

- Text must be grammatically sound and spelled correctly. Poor spelling loses credibility points straight away. Ensure that there is plenty of well laid out textual content on the site to attract search engines as well as to inform prospective clients.
- Use keyword and key phrase rich text; that is, utilize copy that includes common phrases that people would enter into search engines when performing a query.
- **TEAM5OAGS** needs to be viewable from at least IE and Firefox browsers.
- Images are a wonderful medium to assist in the online application, especially useful to those clients with poor literacy levels or who are in a rush, as we all seem to be these days. But remember, while a picture may be worth a thousand words in the offline world, its worth next to nothing when it comes to search engines as spiders do not 'see' pictures.

Image HTML coding should also contain 'alt' tags. This is a textual representation of the image which is useful for the situations where the image doesn't load for

some reason. Search engines spiders also latch on to this content, especially if the image is linked to another page. 'alt' text will also pop up when a visitor moves their mouse over the image. Client requires that pictures and (not required) videos be used in **TEAM5OAGS.**

- **TEAM5OAGS** site navigation should be simple and all the questions a consumer may ask should be answered along the way. Where possible, adhere to the "three click rule" - that is, a visitor should be able to access any information regarding your service within 3 clicks of any other area of your web site.

- **TEAM5OAGS** should take advantage of the Javascript and Ajax technologies.

UC15: **request online chat**

- **TEAM5OAGS** implements a live online support chat allowing communication of

guests and customers with **TEAM5OAGS** operators over the Internet in real time directly from **TEAM5OAGS** web site. No chat information will be stored in the database.

- **TEAM5OAGS** web site will have at least the following pages "**TEAM5OAGS**" home page, "About Us" page, "Contact Us" page, "Testimonials" page with a

UC16: request search

possible search function. Testimonials are stored in the database. The search is done by a stored procedure.

- TEAM5OAGS is a secure web application with a relational database backend (TEAM5OAGS).

UC20: sell art

TEAM5 is a small art gallery that sells contemporary European and North American fine art, including lithographs, high-quality reproduction prints, original paintings and other artwork, and photographs. All of the lithographs, prints, and photos are signed and numbered, and the original art is usually signed. TEAM5 also provides art framing

UC17: request art framing service

services. It creates a custom frame for each artwork (rather than selling standardized, premade frames) and is known for its excellent collection of frame stock.

TEAM5 emphasizes reproduction artworks of European Impressionist, Abstractionist, and Modernist artists such as Wassily Kandinsky and Henri Matisse. For original art, TEAM5 concentrates on Northwest School artists, such as Mark Tobey, Morris Graves,

GuyAnderson, and Paul Horiuchi, and produces shows of contemporary artists who work in the Northwest School tradition or in Northwest Maritime art. The price of new reproduction prints ranges up to $1,000, and prices for contemporary artists range from $500 to $10,000. The price of art from the Northwest School artists varies considerably, depending on the artwork itself. Small pencil, charcoal, or watercolor sketches may sell for as little as $2,000, whereas major works can range from $10,000 to $100,000. Very occasionally, TEAM5 may carry Northwest School art priced up to $500,000, but art priced above $250,000 is more likely to be sold at auction by a major art auction house.

TEAM5 has been in business for 30 years and has one full-time owner, three salespeople, and two workers who make frames, hang art in the gallery, and prepare artwork for shipment.

UC18: request work order report

TEAM5 holds openings and other gallery events to attract customers to the gallery.

TEAM5 owns all of the art that it sells—even sales of contemporary artwork is treated as a purchase by TEAM5 that then is resold to a customer. TEAM5 does not take items on a consignment basis.

The requirements for the TEAM5 application are as follows:

First, both the owner and the salespeople want to keep track of Customers' last and first names, addresses (street, city, state, zip, country), phone numbers (area code and phone number), and e-mail addresses by requesting a Customers Report and want to keep track of Artists' last and first names and nationality by requesting an Artists Report. They also want to know which artists have appeal to which customers by requesting a Customer Artists Preferences Report where the Customer Id is the input to such report.

A1: Owner

A2: Salesperson

UC1: request customer report

UC2: request Artist report

UC3: request customer artist preferences report

The salespeople use this information to determine whom to contact when new art arrives and to personalize verbal and e-mail communications with their customers.

UC4: purchase new art

When TEAM5 purchases new art, data about the artist, the nature of the work, the acquisition date, and the acquisition price are recorded in the database backend. Also, on occasion, TEAM5 repurchases art from a customer and resells it, thus a work may appear in the TEAM5 gallery multiple times.

UC5: repurchase art from customer

When art is repurchased, the artist and work data are not reentered, but the most recent acquisition date and price are recorded.

There is a policy at TEAM5 to set the value of AskingPrice equal either to twice the AcquisitionPrice or to the AcquisitionPrice plus the average net gain for sales of this art

in the past, whichever is greater. An AFTER database trigger **TRANSACTION_AskingPriceInitialValue** is to implement this policy. After declaring program variables, the trigger reads the TRANSACTION table to find out how many TRANSACTION rows exist for this work. Because this is an AFTER trigger, the new TRANSACTION row for the work will have already been inserted. Thus, the count will be one if this is the first time the work has been in the gallery. If so, the new value of SalesPrice is set to twice the AcquisitionPrice.

If the user variable rowCount is greater than one, then the work has been in the gallery before. To compute the average gain for this work, the trigger uses an **ArtistWorkNetView** view to compute SUM(NetProfit) for this work. The sum is placed in the variable sumNetProfit. Notice that the WHERE clause limits the rows to be used in the view to this particular work. The average is then computed by dividing this sum by rowCount minus one.

In addition, when art is sold a TRANSACTION record with the purchase date, acquisition price, date acquired, sales price, asking price, and identity of the purchasing customer are stored in the database backend.

TEAM5 has a special interest in Mexican painters and never discounts the price of their works. Thus, the SalesPrice of a work must always be at least the AskingPrice. To enforce this rule, TEAM5 database has an insert and update trigger TRANSACTION_CheckSalesPrice on TRANSACTION that checks to see if the work is by a Mexican painter. If so, the SalesPrice is checked against the AskingPrice. If it is less than the AskingPrice, the SalesPrice is reset to the AskingPrice. This, of course, must happen when the art work is actually being sold, and the customer charged the full amount!

Salespeople want to examine past purchase data so that they can devote more time to the most active buyers by requesting a Past Purchase Report. They also sometimes use the purchase records to identify the location of artworks they have sold in the past by requesting a Past Purchases Artwork Location Report.

UC6: request past purchase report

UC7: request past purchase artwork location report

TEAM5 wants to hire you to create TEAM5OAGS web application for these requirements.

For marketing purposes, TEAM5 wants its TEAM5OAGS web application to provide a

UC8: request artist and works report

list of artists and works that have appeared in the gallery by requesting an Artist and Works Report. The owner also would like to be able to determine how fast an artist's

UC9: request speed of sale report

work sells and at what sales margin by requesting a Speed of Sale Report given an artist Id. The TEAM5OAGS web application also should display current inventory on a web

UC10: request current inventory report

page that customers can access via the Internet by requesting a Current Inventory Report.

UC19: log in

TEAM5OAGS is a secure web application where owner, sales people, customers have

UC11: modify account

accounts and have the ability to modify their own information (not the USERNAME,

UC12: create account

PASSWORD, EMAIL ADDRESS which will be assigned by the TEAM5OAGS web

A4: DBA

application manager/DBA. DBA has access to New Artist Forms and New Customer

UC13: add new artist

Forms (it should include the artist's interested nationality) to allow them to add new

UC14: add new customer

artists and customers.

The contract states that, at the acceptance test, a scenario of usage of the TEAM5OAGS will be given to you. **Data about customers, art work, artists, etc. will be given to the DBAs for your TEAM5.**

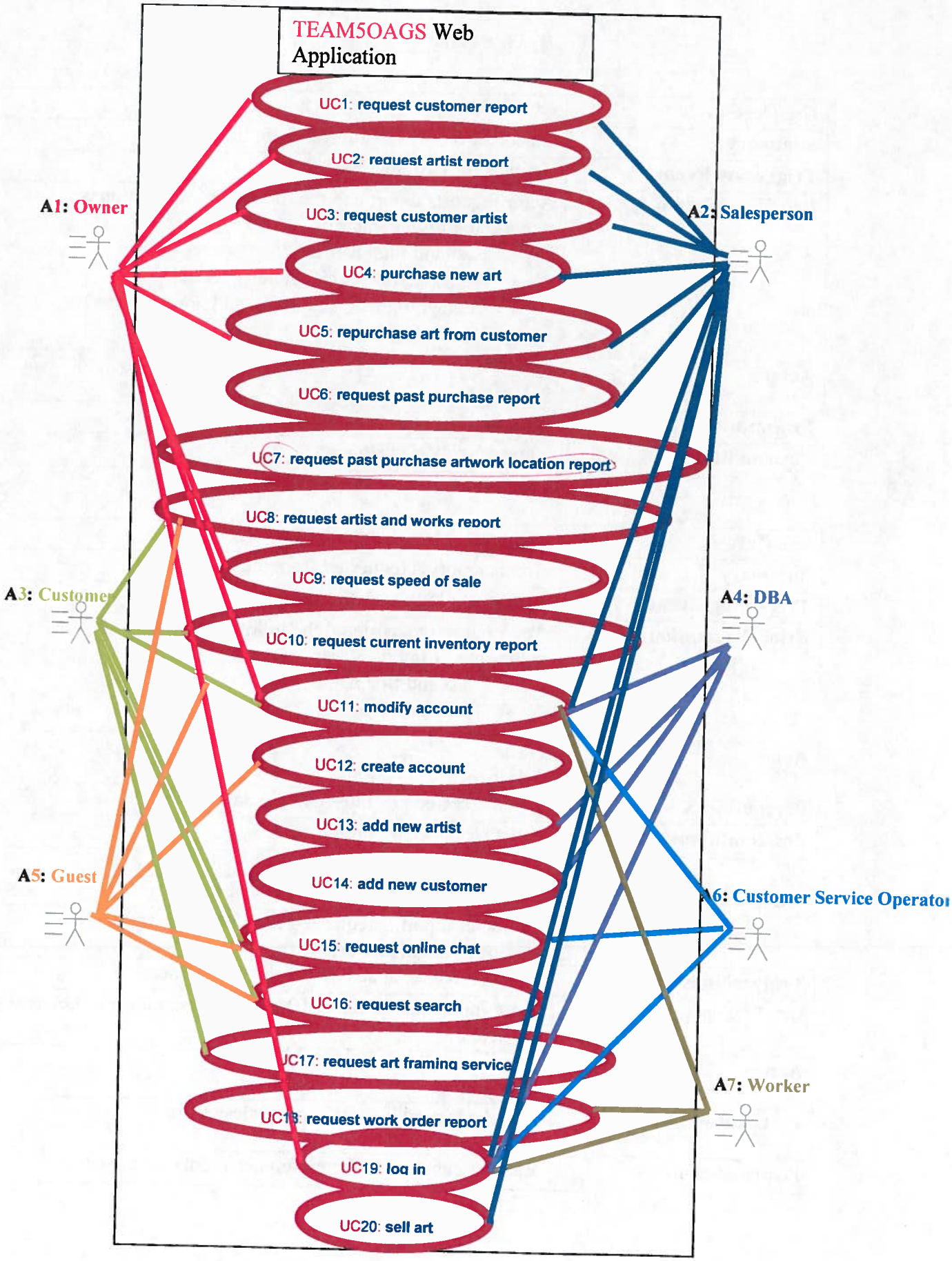
The client desires that TEAM5OAGS be designed and implemented using OO paradigm. Formal Analysis, SRS (Software Requirements Specification) document needs to be signed by the TEAM5 and the client BEFORE any design is started. A SPMP (Software Project Management Plan) document needs to be delivered before the actual development is started.

A prototype is highly desirable. Graphical User Interfaces, Web Site Design, each Page Design can expedite the development process.

Coordinate with the database backend DBAs in clearly separating what is going to be placed in the database and what will be implemented in the database (backend) versus what will be implemented in the TEAM5OAGS web application (front end).

An MVC (3 tier architecture) is required.

NOTE: This requirements document "SE Team Project with Line Numbers.doc" might go through revisions based on your questions. There might be missing requirements; there might be unclear requirements, or conflicting requirements. I can take 5 minutes each class to answer possible questions.


TEAM5OAGS Web Application
UC1: request customer report
UC2: request artist report
UC3: request customer artist
UC4: purchase new art
UC5: repurchase art from customer
UC6: request past purchase report
UC7: request past purchase artwork location report
UC8: request artist and works report
UC9: request speed of sale
UC10: request current inventory report
UC11: modify account
UC12: create account
UC13: add new artist
UC14: add new customer
UC15: request online chat
UC16: request search
UC17: request art framing service
UC18: request work order report
UC19: log in
UC20: sell art
A1: Owner
A2: Salesperson
A3: Customer
A4: DBA
A5: Guest
A6: Customer Service Operator
A7: Worker

UML Use Case Descriptions

Use Case	UC 1 – Request Customers Report
Summary	Customer report is requested from the database
Triggering Event	Request for customer records
Brief Description	Actor requests report and the following information is represented from database • Last and First names • Addresses(street,city,state,zip,country) • Phone numbers(area code and phone number) • Email addresses
Actors	Owner Salesperson
Precondtions	Must be logged in as owner or salesperson
Postconditions	View of request customers report

Use Case	UC 2 – Request Artists Report
Summary	Artists report is requested from the database
Triggering Event	Wanting to look at Artist records
Brief Description	Actor requests report and the following information is represented from database • Last and first names • Nationality
Actors	Owner Salesperson
Precondtions	The Artist Records must exist in database
Postconditions	Artist report view is returned

Use Case	UC 3 – Request customer artist preferences report
Summary	Customer report is requested from the database based on previous bought artwork
Triggering Event	Request to look at customer artist preferences
Brief Description	Actor inputs the customer ID and the customers art preferences are listed
Actors	Owner Salesperson
Precondtions	Must be logged in as owner or salesperson
Postconditions	View or customer artist preference report is retuned

Use Case	UC 4 – Purchase New Art
Summary	A new piece of art is purchased and added to the inventory
Triggering Event	New art is acquired
Brief Description	The following information is input when buying new art. • Data about the artist • Nature of the work • Acquisition date • Acquisition price All input into database
Actors	Owner Salesperson
Precondtions	Must be logged in as owner or salesperson
Postconditions	Purchase new art view is returned

Use Case	UC 5 – Repurchase Art From Customer
Summary	A piece of art is repurchased and added back to the inventory
Triggering Event	New art is acquired
Brief Description	The following information is input when buying back the art. • Data about the artist • Nature of the work • Acquisition date • Acquisition price All input into database
Actors	Owner Salesperson
Precondtions	Must be logged in as owner or salesperson
Postconditions	Repurchased view is returned

Use Case	UC 6 – Request Past Purchase Report
Summary	Report to examine past purchase data so that actor can devote more time to the most active buyers
Triggering Event	Trying to sell more art
Brief Description	A report displaying the past purchases
Actors	Salesperson
Precondtions	Must be logged in as salesperson
Postconditions	Past purchase view is returned

Use Case	UC 7 – Request Past Purchase Artwork Location Report

Summary	To identify the location of artworks they have sold in the past
Triggering Event	Trying to sell more art
Brief Description	A report displaying the location of past purchased artwork
Actors	Salesperson
Precondtions	Must be logged in as salesperson
Postconditions	Past purchase Artwork Location report view is returned

Use Case	UC 8 – Request Artist and Works Report
Summary	To provide a list of artists and works that have appeared in the gallery
Triggering Event	Trying to sell more art
Brief Description	A report displaying each artist and the works he/she has done.
Actors	Customer Guest
Precondtions	Must be logged in as customer or guest
Postconditions	Artist and works view is returned

Use Case	UC 9 – Request Speed of Sale Report
Summary	To determine how fast an artist's work sells and at what sales margin
Triggering Event	Request is made to site
Brief Description	Given and artists id a report is given based on what has sold and how much profit
Actors	Owner
Precondtions	Must be logged in as owner
Postconditions	Speed of sale report view is returned

Use Case	UC 10 – Request Current Inventory Report
Summary	Display current inventory on a web page
Triggering Event	Trying to sell more art
Brief Description	Display current inventory on webpage
Actors	Customer
Precondtions	Must be logged in as customer
Postconditions	Current instance of all inventory is added to inventory view and returned to user

Use Case	UC 11 – Modify Account
Summary	All actors have accounts and have the ability to modify their

	email or password
Triggering Event	Need to change info
Brief Description	Modify view is retuned to change password, email or both.
Actors	Owner SalesPerson Customer
Precondtions	Account info must exist for Actor and Must be logged in
Postconditions	Account info in modified

Use Case	UC 12 – Create Account
Summary	Forms to fill out to assign account information
Triggering Event	New Account information is sent to DBA's view to add new customer or not
Brief Description	All relevant information about account holder in assigned
Actors	Guest
Precondtions	Must not be logged in
Postconditions	Registration view is returned

Use Case	UC 13 – Add New Customer
Summary	New customer information is added to the database
Triggering Event	A new customer is acquired
Brief Description	Customer information is entered into the database • Last and First names • Addresses(street,city,state,zip,country) • Phone numbers(area code and phone number) Email addresses
Actors	DBA
Precondtions	Guest must have already used create an account & Must be logged in as DBA
Postconditions	New customer is added to the database

Use Case	UC 14 – Add New Artist
Summary	New Artist information is added to the database
Triggering Event	A new Artist is acquired
Brief Description	Customer information is entered into the database • Last and first names • Nationality

Actors	DBA
Precondtions	Customer database must exist & Must be logged in as DBA
Postconditions	New customer is added to the database

Use Case	UC 15 – Request Online Chat
Summary	Customer or Guest chats with customer service operator
Triggering Event	Guest or customer request to chat with operators
Brief Description	• Customer service operator chats with guest or customer to answer any questions they might have
Actors	Customer or Guest
Precondtions	Customer database must exist and must be logged in as DBA
Postconditions	Customer or Guest connects with first available customer servie operator

Use Case	UC 16 – Request Search
Summary	Search keys are entered into the search text box and user requests search (Integrated Custom Google Search)
Triggering Event	Customer or guest requests to search for keys in website
Brief Description	• Customer or guest can request a search for particular keys on the website which returns all relevant pages.
Actors	Customer or Guest
Precondtions	Must be logged in as customer or guest
Postconditions	Search results page is returned

Use Case	UC 17 – Request Art Frame Service
Summary	Customer orders frame from the frame stock
Triggering Event	
Brief Description	Customer information is entered into the database • Last and first names • Nationality
Actors	Customer
Precondtions	Customer must be on frame order page and have appropriate input data filled
Postconditions	An order for frame stock is placed with measurements and added to the work order list

Use Case	UC 18 – Request Work Order List
Summary	Worker sees orders that need to be completed
Triggering Event	Worker logged in

Brief Description	• Worker completes current work orders in work order list such as building frames for art orders, hanging art, and shipping new orders.
Actors	Worker
Precondtions	Worker must be logged in
Postconditions	After worker checks that an order is complete it is then removed from the work order list

Use Case	UC 19 – Log in
Summary	Guest logged into website and verifies account
Triggering Event	Username and password is verified
Brief Description	Guest logged into website, username and password is first verified then privileges and home view is returned based on account type
Actors	Guest
Precondtions	Guest must request to log in
Postconditions	If verified account's type view is returned else incorrect password view is returned with chance to register or recovery user name or password

Use Case	UC 20– sell art
Summary	Salesperson uses sell art to process an order for a customer
Triggering Event	Salesperson process order
Brief Description	Salesperson fills out an order form for a customer then hits the process order button.
Actors	Salesperson
Precondtions	Salesperson must be logged in and order form filled out
Postconditions	Order is placed in the system and additional information is sent to work order list

DOCUMENT CONTROL

CHANGE HISTORY

TLs entries (assigned work and due dates) before releasing to the team

Revision	Name	Due Date	Description
1.1	Lee, Ryan	02/01/2013	Identified Actors and Use Cases
1.2	Loucks, Joel	02/01/2013	Complete Use Case Modeling
1.3	Ohlson, Gabriel	02/01/2013	Complete Use Case Modeling
1.X	Cruz, Christopher	02/05/2013	Review Document
1.Y	Naviwala, Muhammad	02/05/2013	Review Document

TMs entries when they completed their work

Revision	Name	**Completed Date**	Description
1.A	Joel Loucks	01/20/2013	Identified actors for use case diagram.
1.B	Ryan Lee	01/20/2013	Identified additional actors and Use Cases.
1.C	Ryan Lee	01/20/2013	Added preliminary UML UC diagram.
1.D	Joel Loucks	01/20/2013	Made actors in diagram a group and added UC# labels to UC diagram.
1.E	Muhammad Naviwala	01/22/2013	Reviewed document. Changed/completed UC diagram. Changed some grammar in sentences.
1.F	Christopher Cruz	01/22/2013	Reviewed, edited some grammar. Added class diagrams with just attributes. Still missing class diagram.
1.G	Joel Loucks	01/24/2013	Added actors and use cases to their respective tables on page 1 of document.
1.H	Ryan Lee	01/24/2013	Finished preliminary text analysis and UC diagram.
1.I	Christopher Cruz	01/24/2013	Cleaned up tables on first page.
1.J	Joel Loucks	01/25/2013	Added use case descriptions for each use case, but still need some info added in some of them. Added line from customer to UC10.

1.K	Joel Loucks	01/28/2013	Updated my work in the change history in the word document.
1.L	Gabriel Ohlson	01/28/2013	Added lots of color and changed some use cases and use case descriptions.
1.M	Christopher Cruz	01/29/2013	Final review and sign.
1.N	Muhammad Naviwala	01/30/2013	Document looks pretty good and the new revision has followed all guidelines and templates. Final review done.
1.O	Gabriel Ohlson	02/01/2013	Found 3 more actors and added 5 more use cases. Updated all use cases and filled out all left over boxes as well as changed them to account for new use cases
1.P	Cruz Christopher	02/01/2013	Fixed some formatting, reviewed. Really looking good now.
1.Q	Muhammad Naviwala	02/05/2013	Modified the aligning of the tables and some formatting. The use case numbering is not in ordering though. Everything looks okay now. Final review done.
1.R	Cruz, Christopher	02/05/2013	Final Review done, looks great.
1.S	Ryan Lee	02/05/2013	Team leader final review and approval for Red Deliverables.

TLs entry before RED TEAMS DELIVERABLES

Revision	Name	Due Date	Description
2.0	Lee, Ryan	02/05/2013	I changed Version to 2.0

DOCUMENT STORAGE

This file is stored in SVN at http://limi.cs.uh.edu/COSC4351/TEAM5/TEAM PROJECT DELIVERABLES/**SE Team Project with Line Numbers for UML USE CASE DIAGRAM.doc**.

TEAM SOAG S (30) OOA class variables 100 points

prev. 60

SE MVC CLASS MODEL (Iteration) TL did not assign work -30

RFD Based on RFD TEAMS DELIVERABLES Feedback

Original #point

(10) "Requirements" or "Multiple Requirements" page (4) (11) pages (147) Line Numbers (48) last line Number

-20 points

Please provide Single Sided

Step 1 (10) Classes prev. 14

UML Compilable (14) ✓ Classes To include

To include C classes (from UML USE CASE MODEL)

Did not use UML USE CASE -20

SAME!

100% "Cut" & "Paste" -30 points

Step 2 Attributes prev. 42

Compilable UML (42) To include CLASS: #

-20 points

(10) ~~Step 4~~ ~~MVC CLASS MODEL (OMFG!)~~ (X) "Cut" & "Paste"

Not in Word (X) -40

100% match (X) -40 points

If not changed at All from feedback -100 points or Ø

T2 ✓ did not assign ✓ Work! (-30)

TEAM5OAGS

Please print single sided!

SE Team Project with Line Numbers
TEXTUAL ANALYSIS for
OOA Workflow
UML CLASS DIAGRAM

# of Classes: 14

Table of Classes:

Class #	Class Name
1	AccountHolder
2	Artist
3	Artwork
4	Customer
5	Transaction
7	Testimonial
8	Frame
9	Inventory
10	ReportController
11	OrderController
12	AccountController
13	DBAController
14	ChatController
15	Model
16	View
17	Controller

# of Attributes: 42

Table of Attributes:

Class Name	Attributes
Artwork	artID natureWork artImage
AccountHolder	last first street city state zip country areaCode phoneNumber email role userID password
Artist	aLast aFirst nationality artistID
Customer	CID
Transaction	acquisitionDate acquisitionPrice purchaseDate salesPrice askingPrice location transID
Testimonial	testID Test_Content Test_Date
Employee	EID
Frame	FID frameType width height

	rabbet color style
Inventory	itemID itemType itemPrice

1

Version 2.0

3.0

Steps 1, 2, 3 - Identify Classes, Attributes, verbs Methods
(Square = Class, Oval = Attributes, Underline = verb Methods)

Team Project: TEAM5OAGS (OnLine Art Gallery System) Web Site

The TEAM5OAGS web site should be designed using these principles:

- Text must be grammatically sound and spelled correctly. Poor spelling loses credibility points straight away. Ensure that there is plenty of well laid out textual content on the site to attract search engines as well as to inform prospective clients.
- Use keyword and key phrase rich text; that is, utilize copy that includes common phrases that people would enter into search engines when performing a query.
- TEAM5OAGS needs to be viewable from at least IE and Firefox browsers.
- Images are a wonderful medium to assist in the online application, especially useful to those clients with poor literacy levels or who are in a rush, as we all seem to be these days. But remember, while a picture may be worth a thousand words in the offline world, its worth next to nothing when it comes to search engines as spiders do not 'see' pictures.

  Image HTML coding should also contain 'alt' tags. This is a textual representation of the image which is useful for the situations where the image doesn't load for

some reason. Search engines spiders also latch on to this content, especially if the image is linked to another page. 'alt' text will also pop up when a visitor moves their mouse over the image. Client requires that pictures and (not required) videos be used in TEAM5OAGS.

- TEAM5OAGS site navigation should be simple and all the questions a consumer may ask should be answered along the way. Where possible, adhere to the "three click rule" - that is, a visitor should be able to access any information regarding your service within 3 clicks of any other area of your web site.

- TEAM5OAGS should take advantage of the Javascript and Ajax technologies.

ChatController: 15 - Method: requestOnlineChat

- TEAM5OAGS implements a live online support chat allowing communication of guests and customers with TEAM5 OAGS operators over the Internet in real time directly from TEAM5OAGS web site. No chat information will be stored in the database.

Customer

- TEAM5OAGS web site will have at least the following pages "TEAM5OAGS" home page, "About Us" page, "Contact Us" page, "Testimonials" page with a

Testimonial

ReportController: 16 - Method: requestSearch

possible search function. Testimonials are stored in the database. The search is done by a stored procedure.

- TEAM5OAGS is a secure web application with a relational database backend (TEAM5OAGS).

TEAM5 is a small art gallery that sells contemporary European and North American fine art, including lithographs, high-quality reproduction prints, original paintings and other artwork, and photographs. All of the lithographs, prints, and photos are signed and numbered and the original art is usually signed. TEAM5 also provides art framing services. It creates a custom frame for each artwork (rather than selling standardized, premade frames) and is known for its excellent collection of frame stock.

Artwork

Artwork: 1 - artID

OrderController: 17 - Method: requestArtFramingService

Frame

TEAM5 emphasizes reproduction artworks of European Impressionist, Abstractionist, and Modernist artists such as Wassily Kandinsky and Henri Matisse. For original art, TEAM5 concentrates on Northwest School artists, such as Mark Tobey, Morris Graves,

Artist

GuyAnderson, and Paul Horiuchi, and produces shows of contemporary artists who work in the Northwest School tradition or in Northwest Maritime art. The price of new reproduction prints ranges up to $1,000, and prices for contemporary artists range from $500 to $10,000. The price of art from the Northwest School artists varies considerably, depending on the artwork itself. Small pencil, charcoal, or watercolor sketches may sell for as little as $2,000, whereas major works can range from $10,000 to $100,000. Very occasionally, TEAM5 may carry Northwest School art priced up to $500,000, but art priced above $250,000 is more likely to be sold at auction by a major art auction house.

Employee

TEAM5 has been in business for 30 years and has one full-time owner, three salespeople and two workers who make frames, hang art in the gallery, and prepare artwork for shipment.

ReportController: 18 - Method: requestWorkOrderReport

TEAM5 holds openings and other gallery events to attract customers to the gallery.

TEAM5 owns all of the art that it sells—even sales of contemporary artwork is treated as a purchase by TEAM5 that then is resold to a customer. TEAM5 does not take items on a consignment basis.

The requirements for the TEAM5 application are as follows:

AccountHolder: 2 - first
AccountHolder: 1 - last

First, both the owner and the salespeople want to keep track of Customers last and first

AccountHolder: 3 - street, 4 - city, 5 - state, 6 - zip, 7 - AccountHolder: 8 - areaCode

names, addresses street city state zip country, phone numbers area code and phone

AccountHolder: 1 - email
AccountHolder: 9 - phoneNumber
ReportController: 1 - Method: requestCustomersReport

number, and e-mail addresses by requesting a Customers Report and want to keep track

Artist: 1 - aLast
Artist: 2 - aFirst
Artist: 3 - nationality
ReportController: 2 - Method: requestArtistsReport

of Artists last and first names and nationality by requesting an Artists Report. They

also want to know which artists have appeal to which customers by requesting a

Customer: 10 – CID
Controller: 3 - Method: requestCustomerArtistsPreferencesReport

Customer Artists Preferences Report where the Customer Id is the input to such report.

The salespeople use this information to determine whom to contact when new art arrives

and to personalize verbal and e-mail communications with their customers.

Artwork: 2 - natureWork
OrderController: 4 - Method: purchaseNewArt

When TEAM5 purchases new art, data about the artist, the nature of the work, the

Transaction: 1 - acquisitionDate
Transaction: 2 - acquisitionPrice

acquisition date, and the acquisition price are recorded in the database backend. Also, on

OrderController: 5 - Method: repurchaseArt

occasion, TEAM5 repurchases art from a customer and resells it, thus a work may appear

in the TEAM5 gallery multiple times.

When art is repurchased, the artist and work data are not reentered, but the most recent

acquisition date and price are recorded.

There is a policy at TEAM5 to set the value of AskingPrice equal either to twice the

AcquisitionPrice or to the AcquisitionPrice plus the average net gain for sales of this art

in the past, whichever is greater. An AFTER database trigger

TRANSACTION_AskingPriceInitialValue is to implement this policy. After declaring

Transaction

program variables, the trigger reads the TRANSACTION table to find out how many

TRANSACTION rows exist for this work. Because this is an AFTER trigger, the new

TRANSACTION rowfor the work will have already been inserted. Thus, the count will

be one if this is the first time the work has been in the gallery. If so, the new value of

SalesPrice is set to twice the AcquisitionPrice.

If the user variable rowCount is greater than one, then the work has been in the gallery

before. To compute the average gain for this work, the trigger uses an

ArtistWorkNetView view to compute SUM(NetProfit) for this work. The sum is placed

in the variable sumNetProfit. Notice that the WHERE clause limits the rows to be used in

the view to this particular work. The average is then computed by dividing this sum by

rowCount minus one.

OrderController: 20 - Method: sellArt Transaction: 3 - purchaseDate

In addition, when art is sold a TRANSACTION record with the purchase date,

Transaction: 4 - salesPrice Transaction: 5 - askingPrice

acquisition price, date acquired sales price asking price and identity of the purchasing

customer are stored in the database backend.

TEAM5 has a special interest in Mexican painters and never discounts the price of their works. Thus, the SalesPrice of a work must always be at least the AskingPrice. To enforce this rule, TEAM5 database has an insert and update trigger TRANSACTION_CheckSalesPrice on TRANSACTION that checks to see if the work is by a Mexican painter. If so, the SalesPrice is checked against the AskingPrice. If it is less than the AskingPrice, the SalesPrice is reset to the AskingPrice. This, of course, must happen when the art work is actually being sold, and the customer charged the full amount!

Salespeople want to examine past purchase data so that they can devote more time to the most active buyers by requesting a Past Purchase Report. They also sometimes use the purchase records to identify the location of artworks they have sold in the past by requesting a Past Purchases Artwork Location Report.

TEAM5 wants to hire you to create TEAM5OAGS web application for these requirements.

For marketing purposes, TEAM5 wants its TEAM5OAGS web application to provide a

ReportController: 8 - Method: requestArtistandWorksReport
list of artists and works that have appeared in the gallery by requesting an Artist and

Works Report. The owner also would like to be able to determine how fast an artist's

ReportController: 9 - Method: requestSpeedofSaleReport
work sells and at what sales margin by requesting a Speed of Sale Report given an artist

Artist: 4 - artistID
Id. The TEAM5OAGS web application also should display current inventory on a web

ReportController: 10 - Method: requestCurrentInventoryReport
page that customers can access via the Internet by requesting a Current Inventory

Report.

AccountController: 19 - Method: loginAccount AccountHolder: 2 - role
TEAM5OAGS is a secure web application wher owner, sales people, customer have
Accountholder
AccountController: 11 - Method: modifyAccount AccountHolder: 3 - userID
accounts and have the ability to modify their own information (not the USERNAME,
AccountHolder: 4 - password DBAController: 12 - Method: createAccount
PASSWORD EMAIL ADDRESS which will be assigned by the TEAM5OAGS web

application manager/DBA. DBA has access to New Artist Forms and New Customer

DBAController: 13 - Method: addNewArtist
Forms (it should include the artist's interested nationality) to allow them to add new

DBAController: 14 - Method: addNewCustomer
artists and customers.

The contract states that, at the acceptance test, a scenario of usage of the TEAM5OAGS will be given to you. **Data about customers, art work, artists, etc. will be given to the DBAs for your TEAM5.**

The client desires that TEAM5OAGS be designed and implemented using OO paradigm. Formal Analysis, SRS (Software Requirements Specification) document needs to be signed by the TEAM5 and the client BEFORE any design is started. A SPMP (Software Project Management Plan) document needs to be delivered before the actual development is started.

A prototype is highly desirable. Graphical User Interfaces, Web Site Design, each Page Design can expedite the development process.

Coordinate with the database backend DBAs in clearly separating what is going to be placed in the database and what will be implemented in the database (backend) versus what will be implemented in the TEAM5OAGS web application (front end).

142

143 An MVC (3 tier architecture) is required.

144 NOTE: This requirements document "SE Team Project with Line Numbers.doc" might

145 go through revisions based on your questions. There might be missing requirements;

146 there might be unclear requirements, or conflicting requirements. I can take 5 minutes

147 each class to answer possible questions.

148 DOCUMENT CONTROL

CHANGE HISTORY

TLs entries (assigned work and due dates) before releasing to the team

Revision	Name	Due Date	Description
1.1	Ryan Lee	02/01/2013	Complete Class Modeling
1.2	Matthew Liang	02/01/2013	Complete Class Modeling
1.3	Yash Jaradi	02/01/2013	Complete Class Modeling
1.X	Christopher Cruz	02/05/2013	Review Document
1.Y	Muhammad Naviwala	02/05/2013	Review Document

Not assigned work!

TMs entries when they completed their work

Revision	Name	Completed Date	Description
1.A	Ryan Lee	01/18/2013	Text analysis update: identified potential objects and attributes for Art, Artist, Customer classes
1.B	Ryan Lee	01/20/2013	Text analysis update: added preliminary UML models
1.C	Christopher Cruz	01/22/2013	Reviewed, edited some grammar. Added class diagrams with just attributes. Still missing class diagram.
1.D	Matthew Liang	01/25/2013	Done TA for steps 1, 2, 3
1.E	Matthew Liang	01/27/2013	Few changes to TA, Added Table Content & UML diagram (step 4)
1.F	Muhammad Naviwala	01/30/2013	Besides the "detailed pseudocode" part, everything looks good. Might need to change template when identifying classes.
1.G	Yash Jaradi	02/01/2013	Cleaned up the TA, changed back to the required templates, found attributes for account holder, added methods for Controller and Transactions.
1.H	Matthew Liang	02/02/2013	Changed TA methods to match Use Case, renumber, bolded text, added class & attributes for frame, reassign classes to attributes
1.I	Yash Jaradi	2/03/2013	Identified possible class Discount and it's attributes and methods and also found 2

Give roles!

			attribute for Transaction which were added to the diagram.
1.J	Matthew Liang	02/03/2013	Added pseudocode for 19 UC methods, Changed attribute Frame_Image to Style, Remove Discount and its property, Reassign rowCount and sumNetProfit to Artwork
1.K	Matthew Liang	02/05/2013	Split the Controller classes into 5 and reassign methods, added Inventory class and its property, add sellArt(), rename a few attribute (look at table), rowCount and sumNetProfit removed.
1.L	Ryan Lee	02/05/2013	Cleaned up the attribute and method tables.
1.M	Yash Jaradi	02/05/2013	Modified the classes according to the new methods for the controller.
1.N	Christopher Cruz	02/05/2013	Connected controllers and views, and reviewed new changes
1.O	Christopher Cruz	02/05/2013	Connected controllers, views, models
1.P	Ryan Lee	02/05/2013	Refined the class diagram in accordance to MVC.
1.Q	Christopher Cruz	02/05/2013	Final Review Done, looks great.
1.R	Muhammad Naviwala	02/05/2013	Modified the formatting of the document a little bit. The class diagram now matches the Table of Attributes. Looks awesome ! Final review done.
1.S	Ryan Lee	02/05/2013	Team leader final review and approval for Red Deliverables
1.T	Christopher Cruz	02/14/2013	Removed Pseudocode, and fixed another error. Signing off.
1.U	Muhammad Naviwala	02/14/2013	Follows the OOA requirements now. Signing off.
1.V	Muhammad Naviwala	03/03/2013	Removed the Table of Methods and UML Class Diagram to make it OOA.

TLs entry before RED TEAMS DELIVERABLES

Revision	Name	Due Date	Description
2.0	Ryan Lee	02/05/2013	I changed Version to 2.0

3.0 3.0

DOCUMENT STORAGE

This file is stored in SVN at http://limi.cs.uh.edu/COSC4351/team5/TEAM PROJECT DELIVERABLES/**SE Team Project with Line Numbers for UML CLASS DIAGRAM.doc**.

TEAM 50 AGS

Ø 100 points

Team Project Report
(Iteration)
Based on RED TEAMS DELIVERABLES
Feedback

Prev. 100

Work not assigned
ch. I, II, III,

Ø

nothing changed!

If not changed at All from feedback
-100 points or Ø

TEAM 50 AGS

65

100 points

SRS

DBAs did not work on this!

Did not remove the blue text

Table 3.2 Requirements

Not ordered

-10 points

F UC1 Functional 20

UC20

NF Non Functional 20

Appendix matches 100% the UML Use CASE MODEL

-10 points

-20 points

UCs 20 # Actors 7

USE CASE MODEL

UC 7 Line Number 10

request part purchase work order

UC Description X UC Diagram ALL? -10 points per team

Not signed by

-10 points

UC 4 complete -√

Not in Word -40

No First Page Summary

(in Word ie. by Hand)

Actors 7

UCs 20

FRs 20 -20

NFRs 20

"Customer" Dr. H has signed SRS? ☒

TEAM5OAGS

Software Requirements Specification (SRS)

# of Actors: 7

# of UCs: 20

Cover sheet UC Model Wrong

# of Functional Requirements: 20

# of Non Functional Requirements: 20

Version 2.0

Approvals Signature Block

COSC 4351	Name	Signature	Date
Customer	Dr. Victoria Hilford		
TL	Ryan Lee		3/4/13
SQA	Christopher Cruz		3/4/13
SQA	Muhammad Naviwala		3/4/13

DOCUMENT CONTROL

CHANGE HISTORY

TLs entries (assigned work and due dates) before releasing to the team

Revision	Name	Due Date	Description
2.1	Ryan Lee	02/22/2013	Complete Sections 1.1 – 1.2
2.2	Matthew Liang	02/22/2013	Complete Sections 1.3 – 1.5
2.3	Christopher Berkowitz	02/22/2013	Complete Section 2
2.4	Yash Jaradi	02/22/2013	Complete Section 3
2.X	Christopher Cruz	02/24/2013	Review Document
2.Y	Muhammad Naviwala	02/24/2013	Review Document
2.Z	Ryan Lee	02/25/2013	Team Leader Final Review and Approval for Purple Deliverables

TMs entries when they completed their work

Revision	Name	**Completed Date**	Description
1.A	Chris Berkowitz	02/21/2013	I completed section 2.0 (Overall Description) more coming soon.
1.B	Chris Berkowitz	02/22/2013	I completed section 2.1 (Product Perspective). More on its way.
1.C	Chris Berkowitz	02/22/2013	I completed section 2.1.1 (System Interface) More on its way.
1.D	Chris Berkowitz	02/22/2013	Completed half of section 2.1.2
1.E	Chris Berkowitz	02/22/2013	Completed remaining section of 2.1.2
1.F	Matthew Liang	02/22/2013	Completed section 1.3 to 1.5
1.G	Yash Jaradi	02/22/2013	Completed sections 3.1,3.5,3.3.6.4,3.7.5 More on the way.
1.H	Muhammad Naviwala	02/24/2013	Read through the "completed" sections/sub-sections. Corrected some grammar and semantics. Added more acronyms. For chapter 3 please use the <Click Here to Insert Your Text> boxes…
1.I	Christopher Cruz	02/24/2013	Read completed sections. Edited Grammar and flow of some sentences. Still missing quite a

			bit.
1.J	Chris Berkowitz	02/27/2013	Added more chapter 2. More incoming.
1.K	Chris Berkowitz	02/27/2013	Added 2.1.4.
1.L	Chris Berkowitz	02/27/2013	Added 2.1.8.
1.M	Chris Berkowitz	02/27/2013	Added 2.3.
1.N	Yash Jaradi	02/28/2013	Added or modified 3.3, 3.5, 3.5.1, 3.6, 3.6.3, 3.7.5
1.O	Yash Jaradi	02/28/2013	Added the appendix and formatted according to the template provided.
1.P	Muhammad Naviwala	03/02/2013	Read through the newly added content. Added some more abbreviations. Made some sentence flow changes. Might need to change section 3.7.5
1.Q	Christopher Cruz	03/03/2013	Read through new sections, corrected grammar and sentence layout. Changed some document formatting to look better. Still a lot missing.
1.R	Ryan Lee	03/03/2013	Finished sections 1.1 – 1.2
1.S	Yash Jaradi	03/03/2013	Added 3.6.1, 3.6.2 and 3.7.4 rest should be uploaded by tonight
1.T	Christopher Cruz	03/03/2013	Reviewed new content 1.1 – 1.2, looks great. Still Missing parts of 2 & 3.
1.U	Christopher Cruz	03/03/2013	Reviewed 3.6.1, 3.6.2, 3.7.4. corrected grammar and sentence structure
1.V	Matthew Liang	03/03/2013	Added 30% of Section 3.2 (Read 1-40 pages in Line Doc)
1.W	Ryan Lee	03/03/2013	Completed sections 2.2 and 2.5
1.X	Yash Jaradi	03/03/2013	Completed 3.7.1, 3.7.2, 3.7.3,3.7.6,3.7.7,3.7.8, 3.8
1.Y	Christopher Cruz	03/03/2013	Corrected Grammar, sentence structure. Looking good, still missing a couple parts.
1.Z	Muhammad Naviwala	03/03/2013	Read through the whole documents again. Made some sentence flow changes and minor grammar changes. Modified the formatting a little bit.
1.AA	Chris Berkowitz	03/04/2013	Added Sections 2.1.5, 2.1.6, and 2.6. Tomorrow I will complete

			the final two sections.
1.AB	Yash Jaradi	03/04/2013	Added section 3.7
1.AC	Christopher Cruz	03/04/2013	Reviewed and edited 3.7
1.AD	Christopher Cruz	03/04/2013	Looks great. Signing off.
1.AE	Muhammad Naviwala	03/04/2013	Reviewed the new sections. Looks fine. Signing off.
1.AF	Yash Jaradi	03/04/2013	Added section 3.4
1.AG	Matthew Liang	03/04/2013	Completed Section 3.2 (20 Functions , 20 non-Functions)
1.AH	Chris Berkowitz	03/04/2013	Added sections 2.1.7
1.AI	Chris Berkowitz	03/04/2013	Added section 2.4. Thus completing section 2.
1.AJ	Matthew Liang	03/04/2013	Removed Blue Italics and renamed 3.2

TLs entry before PURPLE TEAMS DELIVERABLES

Revision	Name	Due Date	Description
2.0	Ryan Lee	02/25/2013	I changed Version to 2.0

DOCUMENT STORAGE

This file is stored in SVN at http://limi.cs.uh.edu/COSC4351/team5/TEAM PROJECT DELIVERABLES/**Software Requirements Specification.doc**.

The document in this file is adapted from the IEEE standards for Software Project Requirements Specifications, 830-1998, which conforms to the requirements of ISO standard 12207 Software Life Cycle Processes. Tailor as appropriate.

Items that are intended to stay in as part of your document are in **bold**; *blue italic text is used for explanatory information that should be removed when the template is used.*

Table of Contents

new page?

1. INTRODUCTION

The Software Requirements Specification document shall cover in detail the following topics: the introductory background, overall description, specific requirements, and additional information of the system to be developed.

1.1 Purpose

The purpose of the Software Requirements Specification document is to describe the behavior and functionalities of the software product to be developed. It will also describe in detail the non-functional requirements that impose constraints on the design or implementation of the system.

The Software Requirements Specification document is targeted toward the users and developers of the TEAM5OAGS web application. The intended goal in describing the contents of this document is to ensure that both users and developers understand each other's requirements.

1.2 Scope

The software product to be developed is an online art gallery system called TEAM5OAGS. The application to be developed will allow visitors and potential customers to not only view artworks submitted by individual artists, but also give them the ability to reserve artworks for purchase. The application will also enable employees of the art gallery to manage their inventories of artworks, keep track of past customer purchases, communicate directly to customers and site visitors, and manage notifications for the packaging and shipping of artworks.

Overall this document will describe the most important features that will be made available as well as their specifications and constraints. This document will not describe in detail the development environments, languages, and tools to be used in the design and development of the software product.

1.3 Definitions, Acronyms, and Abbreviations

Term or Acronym	Definition
TEAM5OAGS	The Online Art Gallery System Web Application
TEAM5	The Client of the Project
SRS	Software Requirements Specification
D.R.Y	Don't Repeat Yourself
MVC	Model View Controller
DBA	Database Administrator
OOA	Object Oriented Analysis
OOD	Object Oriented Design

Table 1. Definitions and Acronyms

UML

1.4 References

Team5OAGS. *Team Project Report.* COSC 4351 Software Engineering, 2013.

http://limi.cs.uh.edu/COSC4351/team5/ TEAM PROJECT DELIVERABLE

- SE Team Project with L M for UML USE CASE

1.5 Overview

The remainder of the software requirement specification document contains the complete description of how Team5OAGS should behave, including three sections and appendixes. Section 2 covers the physical overview of the product describing the details of the interfaces that will be designed and implemented. Section 3 provides the requirement specifications consisting of the details of the functionality of the product. Section 4 includes the supporting information of the SRS document.

2. Overall Description

TEAM5 is an art company which sells multiple types of artwork and also offers custom framing options. TEAM5 decided to expand their business by adding a new outlet to expand many of their core business processes. To accomplish this, TEAM5OAGS was the software product idea that best aggregated all of the required functionality. TEAM5 had been missing out on a potentially large source of revenue in which many competitors have now accessed to gain an advantage. TEAM5OAGS will connect TEAM5 to a market share that they previously had no access to via the internet. TEAM5 will benefit from this in other ways as well. They can use different types of data including customer data, product data, and sales data to better analyze their business model via superior trend analysis. The culmination of TEAM5's requirements, ideas, and desire to become more successful as a whole is realized in TEAM5OAGS.

2.1 Product Perspective

TEAM50AGS is a standard web application that adheres to proper MVC format in attempt to maintain the highest chance of reusability if TEAM5 decides to upgrade the application in the future. TEAM5OAGS spent a majority of its total development process in the planning phase to ensure that all of its functionality was tested thoroughly and that each module was loosely coupled and completely D.R.Y. The structure or format of TEAM5OAGS shares many similarities with the best applications on the market. Just as many web applications attempt to perfect the MVC format, TEAM5OAGS was refined until the interactions between the models, view, and controller were accurate.

2.1.1 System Interfaces

Accounts – All actors involved with TEAM5OAGS will access this system interface in some form during interaction with the application. Varying levels of accounts will be defined by the modules that are accessible to that specific account type. TEAM5OAGS will go live with seven different account types including the temporary (per-visit) accounts for guests whom have registered with TEAM5OAGS.

Inventory – TEAM5OAGS will also have an inventory system interface to maintain all of the products in which TEAM5 has available or may soon replenish at any given time. Within this interface users will be able access the product currently on hand, potentially make purchases or sales, add new artists to inventory lists, and ensure inventory levels are maintained to some degree. This new inventory system will help TEAM5 improve their accuracy in regards to their source of revenue and other vital accounting forums.

Services – The Services system interface is an important interface that will launch TEAM5 ahead of most of their competitors. One of the basic services that users can access is search functionality for TEAM5OAGS. There are a plethora of analytical services that this system interface entails. For the higher level users such as DBA, different employee levels, and the owner, there are many different reporting options that can be accessed and assessed to help optimize the business processes to their full potential. There are also services that lower level users, such as customers, that can be customized and purchased.

2.1.2 User Interfaces

External Users (Customers) - The External Users interface (website) of TEAM5OAGS is an extremely important facet of the application. With other user interfaces for employees allowing internal access to core business processes and the logic of the application, the external user interface is the presentation to the most important group of users involved: The Customer. This is the end point where many external users will access, peruse, purchase art or services, and sell art back to TEAM5. Understanding the functionality of the web application and the positive impact it will have is key to perfecting the user interface. Therefore, the user interface has been crafted to perfection with customer satisfaction being the main driving force behind its design. First and foremost, correctness in all things such as spelling and grammar are a must. TEAM5OAGS adds a new domain to TEAM5 in which many people may visit without any prior knowledge of the business. Knowing correctness has been emphasized, ensures that TEAM5 will not lose potential revenue due to sloppiness. Another aspect of this interface is the importance that a majority of the users will find the visual aesthetics of the application pleasing rather than loud or obnoxious. Complexity is also incredibly important in the applications design. When users navigate the site they will have a menu accessible at all times and the "three click rule" was maintained throughout most of the website. Combining those aspects and having pictures available for the artwork has created a website in which users will appreciate once TEAM5OAGS is unveiled.

Internal User Interface – The internal user interface is very important for TEAM5OAGS as well. A large portion of this interface is the External User Interface discussed previously. The Internal User Interface allows for increased functionality and therefore is only be accessible to the TEAM5 users. TEAM5 will use the internal user interface to maintain the site. They will be able to chat with customers when necessary and help said customers whenever feasible. A huge difference in the interfaces is the

information that TEAM5 can access and analyze. Employees and the owners will be able to analyze aggregate data ranging from reports of past purchases by individuals to reports on inventory levels, live work order reports, and even information about the artists involved. This interface meets the requirements determined by TEAM5 which will allow them to work more efficiently in an environment that they helped design and are comfortable with.

2.1.3 Hardware Interfaces

TEAM5OAGS will consist of three major components. There will be a database that will be hosted on limi.cs.uh.edu. This database will hold all of the information for the web application. The actual web application (website) will also be hosted on the limi servers and is the second major component. The third component will be the external users. Any able body Internet user will be able to use their personal computer hardware and connect themselves to the application. Any computer with Windows XP, Vista, 7, and 8 will be supported. OSX and any smart phone will also be able to connect.

2.1.4 Software Interfaces

There are many different software products that are used in some way with TEAM5OAGS. From an external viewpoint, User will be able to interact with the web application via many different browser options. Firefox will be the recommended browser and version 15 and greater will work as intended. Google chrome will also be compatible with versions 20 and up. Internet Explorer will be compatible with the latest release only. The web application was written in Ruby following the Rails framework.

2.1.5 Communications Interfaces

The communications Interfaces will be limited to only HTTP for TEAM5OAGS. This means that external users will be able to connect to the internet in whatever way they can, and then be able to connect to the web application through port 80. TEAM5 decided not to allow SSL through port 443 until future efforts can test the necessity of this. Internal users will be able to connect to the application via the localhost. Users need the minimum of a cable or DSL modem to access the application without error.

2.1.6 Memory Constraints

TEAM5OAGS will require the external user have at least a recommended 1.5 GB ram. This will ensure the users can access the content without any memory errors. The application will need at least 8 GB of ram to maintain the stability of the server. This will be monitored closely and if traffic exceeds expectations there is room to add up to 12 GB of ram to make a total of 20GB running the server.

2.1.7 Operations

External User:
TEAM5OAGS has a very limited number of unique operations. The external users will be able to access the website in as a transaction operation. This will allow the external user

to browse the web application and use features such as chat. They can also make selections of items that they wish to attain.
Employee:
The employees will have a slightly different mode of operation. They will be able to do all of the operations that an external user has access to. They will also be able to modify the inventory tracking system to maintain proper levels of select goods. The DBA will have access to the database backend which will control the data sets involved in TEAM5OAGS. The owner will be able to do all of the above. All employees will be able to select reports and have them printed at will. The web application will generally be unattended and will continue to grant access in this state.

2.1.8 Site Adaptation Requirements

TEAM5OAGS was written in a way that favors change. Since TEAM5OAGS is a web application it was planned out in advance that the design must be conducive to the ever changing environment it will be exposed to. When external users access the application and perform actions such as make purchases and create or modify accounts, this will inadvertently change the data stored in the database. The only requirement for this type of modification is that their inputs be checked before processing to prevent unwanted access. Employees will be able to log into their accounts to make changes as well. Any major changes to the layout of the GUI or to the back-end database will require the site to be taken down for major maintenance.

2.2 Product Functions

TEAM5OAGS is an online art gallery web application that is designed to provide utility to the client. The major functions of the TEAM5OAGS application are described below:

Maintain an Inventory of Artworks
TEAM5OAGS shall maintain an inventory of all artworks that are currently being held on display at the art gallery. All artworks in the inventory, as well as the details concerning it, can be viewed through a web gallery as well as individually.

Maintain an Inventory of Artists
An inventory of artists whose works are on display online shall be maintained by the TEAM5OAGS application. Relevant details such as the artist's name, nationality, and associated works in the gallery shall be maintained.

Maintain a Record of Past Purchases
A record of past purchases made by customers shall be maintained. Each transaction conducted where a customer decides to purchase an artwork shall be recorded into the past purchases record and the relevant customer and artwork information are stored as well.

Maintain a Record of Sale Speeds
A record of sale speeds shall be maintained to gauge the frequency of commercial sales occurring within the client's business. Each record shall contain information such as the artwork being sold, the date it was placed in the gallery, and the date it was sold.

Provide User Accounts to Customers and Employees
Customers and employees of the art gallery shall have the option to create user accounts affiliated with the site. These accounts can be used for facilitating quicker communication between parties, maintaining customizable artist preferences, email communication, carrying out transactions, and posting comments to pages.

Provide an Online Chat System for Visitors
TEAM5OAGS shall have an interactive online chat system incorporated to help facilitate communication between visitors and the art gallery employee staff. Visitors who enter messages in the chat text area will have their messages forwarded to an available staff member for responses. During non-operating hours, messages will be saved and forwarded when the operating hours resume.

2.3 User Characteristics

TEAM5OAGS has a wide range of targeted users. All of the external users can be clumped together. This user category requires minimal knowledge in computers and the internet. If they can navigate to the applications via the website it is hosted, then they can use the application. This type of user will have an age requirement of 18 and up. The employee characteristics are a bit more refined. Each employee will have to understand the web design and how to make minor changes to the layout. Certain employees will need knowledge in maintaining a database as well. Proper training in how to analyze the different reports will be important.

2.4 Constraints

TEAM5OAGS will not have any constraints as it pertains to the features derived from the requirements document. The application will go live with all of the features requested.

2.5 Assumptions and Dependencies

The following is a list of assumptions and dependencies for the TEAM5OAGS web application:

- The application shall be viewable on Internet Explorer, Mozilla Firefox, and Google Chrome web browsers.
- The operating environments TEAM5OAGS shall run under include Windows, Mac OS, and Linux.

- The database backend shall utilize checkpointing to withstand total data loss in the event of a product failure.
- The application will utilize Rails version 2.3.8 as the web development framework.

2.6 Apportioning of Requirements

At this point there have been no requirements outlined for TEAM5OAGS that will be delayed. There are multiple extra additions that TEAM5OAGS will be able to easily implement given positive future gain from the initial launch of the application. These additions include the possible additions of RAM and SSL access to the application.

3. Specific Requirements

The TEAM5OAGS website is required to be uncomplicated and manageable and is to be compatible with Internet Explorer and Firefox browsers. TEAM5OAGS will utilize Ruby on Rails and MySQL technologies for front- end and database back-end respectively.

3.1 External Interfaces

There will be several interfaces for TEAM5OAGS which include a System interface, User interface, Software interface, and a Communication interface.

System Interface will consist of 7 different modules respected to each Actor specified by the Use CASES. The modules must have a different view and the functionality allowed for each actor.

Software Interface: TEM5OAGS should only be accessible through a web based application, specifically the browsers Internet Explorer and Firefox. The TEAM5OAGS must be built with Ruby on Rails with a Model View Controller (MVC) format, and MySQL for the database back-end.

User Interface is required to be clean, uncomplicated and easy to maintain. The UI should also practice the "Three Click Rule". TEAM5OAGS is also required to have a different view for each module depending on the Actor and its functionality.

The UI must have the following accessible to all users without the need of any credentials.

Site Navigation:

- Must be clean and simple for any user to navigate through the site.
- Should follow the "Three Click Rule".

Log In page:

- Must have User Name and Password input.
- All types of users must be able to log in which gives them different functionality.

Inventory Page:

- All users must be able to view inventory without the need of logging in.

Search:

- All users should be able to search for specific artwork or Testimonials.

Chat:

- Users must be able to chat with operator/customer service representatives.

3.2 FUNCTIONS

ID	Detail	Type	Priority	Line Numbers	Use Case Name
R1	The TEAM5OAGS shall have the text sound grammatically and spelled correctly	User Interface Non-Functional	M	5	
R2	The TEAM5OAGS shall utilize common phrases to be included in the search engine when performing a query	User Interface Non-Functional	C	9-10	
R3	The TEAM5OAGS shall be viewable in browsers such as IE and Firefox	User Interface Non-Functional	M	11	
R4	The TEAM5OAGS shall include 'alt' tags in image HTML coding	User Interface Non-Functional	M	17	
R5	The TEAM5OAGS shall include pictures and (not required) videos	User Interface Non-Functional	M	21	
R6	The TEAM5OAGS shall adhere to the "three click rule"	User Interface Non-Functional	S	24-25	
R7	The TEAM5OAGS shall take advantage of the Javascript and Ajax technologies	User Interface Non-Functional	M	27	
R8	The TEAM5OAGS shall provide a live online chat support to allow chat between customer/guest and customer service operator	Chat Functional	M	28-30	UC15
R9	The TEAM5OAGS shall have at least: "Home" page, "About Us" page, "Contact Us" page, "Testimonials"	Design Non-Functional	M	32-34	
R10	The TEAM5OAGS shall implement a search function by a stored procedure	Search, Database Functional	M	34-35	UC16
R11	The TEAM5OAGS shall be a secure web application with relational database backend	Security Non-Functional	M	36	
R12	The TEAM5OAGS shall allow Salesperson to sell art and process	Sell Art Functional	M	40	UC20

	an order for a customer				
R13	The TEAM5OAGS shall allow customers to request art framing service	Frame Service Functional	M	43-44	UC17
R14	The TEAM5OAGS shall allow workers to request work order report	View Report Functional	M	58-59	UC18
R15	The TEAM5OAGS shall allow owner & salesperson to request customer report	View Report Functional	M	67	UC1
R16	The TEAM5OAGS shall allow owner & salesperson to request a view of artist report	View Report Functional	M	68	UC2
R17	The TEAM5OAGS shall allow owner & salesperson to request customer artist preferences report	View Report Functional	M	69-70	UC3
R18	The TEAM5OAGS shall allow purchase of new art	Purchase Art Functional	M	73	UC4
R19	The TEAM5OAGS shall allow repurchasing art from customer	Repurchase Art Functional	M	75	UC5
R20	The TEAM5OAGS shall allow a work to appear in the gallery multiple times	Database Non-Functional	M	75-76	
R21	The TEAM5OAGS shall update most recent acquisition date and price when art is repurchased	Database Non-Functional	M	77-78	
R22	The TEAM5OAGS shall ensure the asking price equals to the max of twice the acquisition price or the acquisition price plus the average net gain for sales of this art in the past. (After Database Trigger)	Database Non-Functional	M	79-81	
R23	After asking price is declared, the TEAM5OAGS shall allow a database trigger activate to find out how many TRANSACTION rows exist for a work.	Database Non-Functional	M	82-84	
R24	If an artwork has been in the gallery once, the TEAM5OAGS shall ensure Sales Price is set to twice the Acquisition Price. (After Database Trigger)	Database Non-Functional	M	85-87	
R25	The TEAM5OAGS shall allow a trigger to create a View called ArtistWorkNetView to compute the average gain for a work.	Database Non-Functional	M	89-90	
R26	The TEAM5OAGS shall have insert and update trigger to ensure that an artwork made by a Mexican Painter never gets a discount on the price	Database Non-Functional	M	97-104	

	of their works				
R27	The TEAM5OAGS shall allow Salesperson to request past purchase report	View Report Functional	M	106-107	UC6
R28	The TEAM5OAGS shall allow Salesperson to request past purchase artwork location report	View Report Functional	M	107-109	UC7
R29	The TEAM5OAGS shall allow request artist and works report for marketing purposes	View Report Functional	M	112-114	UC8
R30	The TEAM5OAGS shall allow Owner to request speed of sale report	View Report Functional	M	114-116	UC9
R31	The TEAM5OAGS shall allow Customers to request current inventory report	View Report Functional	M	116-118	UC10
R32	The TEAM5OAGS shall allow owner, sales people, customers, DBA, Workers, and Customer Service Operator to authenticate themselves as users of the system	Authentication Functional	M	119	UC19
R33	The TEAM5OAGS shall allow account users to modify their account's information	Modify Account Functional	M	120	UC11
R34	The TEAM5OAGS shall allow guest to create a new account	Create Account Functional	M	121	UC12
R35	The TEAM5OAGS shall allow DBA to have access to New Artist Forms to add new artist	Add Artist Functional	M	122-124	UC13
R36	The TEAM5OAGS shall allow DBA to have access to New Customer Forms to add new artist	Add Customer Functional	M	122-124	UC14
R37	The TEAM5OAGS shall be designed and implemented using OO paradigm	Design Non-Functional	M	130	
R38	Formal Analysis, SRS document shall be signed by the TEAM 5 and the client before any design is started	Client Requirement Non-Functional	M	131-132	
R39	A SPMP document shall be delivered before the actual development is started.	Client Requirement Non-Functional	M	132-134	
R40	The TEAM5OAGS shall be designed in MVC (3 tier architecture)	Design Non-Functional	M	143	

3.3 Performance Requirements

TEAM5OAGS should be able to handle high amount of traffic during peak hours, and should also be able to handle at least 50,000 registered customer accounts. TEAM5OAGS

does will not have any requirements on any transaction as it does not deal with any sale of art work, it does however require processes such as user log in and retrieving reports that the user may request to be done under 3 seconds at most.

3.4 Logical Database Requirements

The following database requirements contain the access capabilities, functions and relationships.

Table 1. Data Requirements

ID	Detail	Entity Name	Access type (r,w,query)	Line Numbers
D1	Testimonial entity set shall have the following attributes: 1. Testimonial_ID 2. Testimonial_Content 3. Testimonial_Created	E1: Testimonial	 r,query r,w,query r,query	 Ad1 33 33 33

One Table

ID	Detail	Entity Name	Access type (r,w,query)	Line Numbers
D2	Customer Entity set shall have the following attribute: 1. Customer_ID 2. Last_Name 3. First_Name 4. Street 5. City 6. State 7. Zip 8. Country 9. areaCode 10. phoneNumber 11. Email 12. userName 13. passwod	E2: Customer	 r,query r,query r,query r,query r,query r,w,query r,w,query r,w,query r,w,query r,w,query r,w,query r,w,query w(DBA)	 67 64 62 62 63 63 63 63 63 64

ID	Detail	Entity Name	Access type (r,w,query)	Line Numbers
D3	Artist Entity set shall have the following 4 attributes: 1. Artist_ID 2. Last_Name 3. First_Name 4. Nationality 5. DateOfBirth 6. DateDeceased	E3: Artist	 r,query r,w,query r,w,query r,w,query r,w,query r,w,query	 64 65 65 65 Ad3 Ad3

ID	Detail	Entity Name	Access type (r,w,query)	Line Numbers
D4	Art entity set shall have the following 1 attributes: 1. 2. 3. 4. 5.	E4: Works	 r,query r,w,query r,w,query r,w,query r,w,query	 70 70 71 71 76 Ad3

r

ID	Detail	Entity Name	Access type (r,w,query)	Line Numbers
D5	Transaction entity set shall have the following 1 attributes: 1. Transaction_ID 2. Acquisition_Price 3. Date_Acquired 4. Sales_Price 5. Asking_Price 6. DateSold	E5: Transaction	 r,query r,w,query r,w,query r,w,query r,w,query r,w,query	 91 92 92 92 92 92

ID	Detail	Relationship Name	Connects Entity Sets	Cardinality	Line Numbers
D7	Gives relationship set	R1: Gives	Customer & Testimonial	1 M	31

ID	Detail	Relationship Name	Connects Entity Sets	Cardinality	Line Numbers
D8	About relationship set	R2: About	Artist & Art	1 M	70

ID	Detail	Relationship Name	Connects Entity Sets	Cardinality	Line Numbers
D9	Interest relationship set	R3: Interest	Customer & Artist	M 1	67

ID	Detail	Relationship Name	Connects Entity Sets	Cardinality	Line Numbers
D10	Purchase relationship set	R4: Purchase	Customer & Art	M M	91

3.5 Design Constraints

-TEAM5OAGS must be written in Ruby on Rails
-The website should be able to operate on Windows and Mac OS.
-Must use Tortoise SVN as a version control system.

NetBeans 6.9.1.
Table 2. Constraints
or just leave them to DRS!

3.5.1 Standards Compliance.

TEAM5OAGS is required to use Microsoft word to perform Object Oriented Analysis (OOA) and Object Oriented Design (OOD). The OOA methods must be used to analyze the software requirements given by the client in the form UML Class diagram, Use Case diagram, and Pseudo code. TEAM5OAGS will create a conceptual model from the OOA which will include functional constraint such as architecture, and nonfunctional constraints such as reliability, portability and response time.

TEAM5OAGS must use Ruby on Rails framework to create the website along with MySQL as the back-end relational database system. TEAM5OAGS must use the Model View Controller (MVC) format that will have separate modules for each user. The MVC format will also promote synergy among the team members by separating duties.

3.6 Software System Attributes

The success of TEAM5OAGS by delivery time depends on the following Software System Requirements: Reliability, Availability, Security, and Maintainability.

3.6.1 Reliability

The reliability of a product working efficiently is a key factor that determines the quality and usability of the software product. TEAM5OAGS is to concentrate their efforts to make the website as reliable as possible without factoring in external faults that causes failures such as hardware, or natural disasters. To guarantee the reliability of the product, TEAM5OAGS must do acceptance testing on all functional parameters of the website and also choose a framework that is proven to be reliable on different hardware and software.

3.6.2 Availability

TEAM5OAGs should be available at all times of the day, except the hours that maintenance is being performed. The availability of the website highly depends on the servers that host the website and database of TEAM5OAGS. The host servers should be running at all times with minimal failures. In case of downtime the database must be backed up periodically to minimize the loss of data and speed up the recovery process.

3.6.3 Security

Security is one of the key factors for TEAM5OAGS software system requirements as it will be a secure website. To ensure the secureness TEAM5OAGS will use separate modules for each user that limits the functionality of the user depending on their role. TEAM5OAGS is a secure website which will use https to transmit information, it is also required to use encryption algorithm for passwords and sensitive information of the user. The use of MVC format of having separate modules will also restrict functionality for users that are not assigned to the account.

3.6.4 Maintainability

TEAM5OAGS requires a high level of maintainability by using the MVC structures for separate accessibility and functionality of each user. The MVC format will ensure the code being uniform and will reduce the time for maintenance as each type of module will have separate tasks and will not affect other modules if changes are made.

3.6.5 Portability

Portability of software product is a key factor that ensures commercial success of a software product. The portability of software across different hardware and operating system depends on several factors: language the code is written, use of a particular operating system, percentage of components which is host-dependent code, and percentage of code that is host-dependent.

TEAM5OAGS must maintain the highest level of portability that complies with most known types of hardware and software used to host or access the website. The use of Ruby on Rails framework which is known for its portability will ensure TEAM5OAGS easy to maintain and transfer over different architecture. The use of separate modules by following the MVC format will also factor in ease of transferability of code into another language such as PHP, or C#.

ID	Characteristic	Rank
1	Correctness	High
2	Efficiency	High
3	Flexibility	Medium
4	Portability	High
5	Usability	High
6	Maintainability	Medium
7	Security	High
8	Standards Compliance	High
9	Availability	Medium
10	Interoperability	Low

3.7 Organizing the Specific Requirements

TEAM5OAGS has several specific forms of recruitments for organizing it's system. The details of the different form so organizations listed below are in the following in subsections.

-System Mode
-User Class
-Objects
-Use Cases
-Feature
-Stimulus
-Response
-Functional Hierarchy

3.7.1 System Mode

TEAM5OAGS runs on 7 separate modes, and each mode corresponds with the type of user. Depending on what type of user is accessing the webpage, he/she is able to access certain types of functions/methods that may not be available in another mode.

3.7.2 User Class

TEAM5OAGS follows a system that provides different sets of functions to different classes of users. In this case TEAM5OAGS has 7 different types of users and they are Owner, Salesperson, Customer, DBA, Guest, Customer Service Operator, and Worker. Each of these users has their own class of controller, and view. These control classes help restrict or enhance the capabilities a user can have on the system.

3.7.3 Objects

The objects of TEAM5OAGS that are real-world entities which have counterpart within the system are the Artist, Artwork, Customer, Frame, Transaction, and Account Holder. Each of these objects has a separate set of attributes and processes performed by the object.

3.7.4 Use Cases

The Use cases are steps that define the main functionality of the program and serve as guidelines the developer must complete in order to have a successful product.

See the Appendix A.1 for a detailed Use Case model and description.

3.7.5 Feature

TEAM5OAGS must have a chat feature that allows the customer to be able to communicate with a customer service representative. Since TEAM5OAGS does not currently deal with online sales, it will not have features such as a shopping cart, placing an order for the artwork, or selling the artwork. There will remain a possibility of additional features added after delivery of the software depending on the client's request which will be handled in the post-delivery maintenance of the software.

3.7.6 Stimulus

In terms of a system that deals with a Stimulus, TEAM5OAGS has 2 separate processes that influences the rest of the system. TEAM5OAGS requirements to log in to the system is able to influence what type of user can access different functions. The other stimuli is the adding of a customer or an artist to the database by a Database-Administrator.

3.7.7 Response

TEAM5OAGS' processes are organized into sections that correspond with a particular type of action. The Classes listed below have functions that deal with only the instance of a set objects.

Report Controller: all functions regarding reports.
Account Controller: deal with all the account related methods.
DBA Controller: only deals with functions relating to the database.

3.7.8 Functional Hierarchy

TEAM5OAGS does not have a Functional Hierarchy between its classes or methods.

3.8 Additional Comments

There are no additional comments.

4. Supporting Information.

Tables on the following pages provide alternate ways to structure section 3 on the specific requirements.

Appendices

A.1 Outline for SRS Section 3: Organized by use case

- see the SRS Appendix A.1 Outline for SRS Section 3 Organized by Use Case.doc for the format

Table of Actors:

Actor #	Actor	Brief semantics
A1	Owner	Owns TEAM5OAGS.
A2	Salesperson	Handles all sales of TEAM5OAGS.
A3	Customer	The potential buyer of artwork from TEAM5OAGS
A4	DBA	Manages all the database related entries for TEAM4OAGS.
A5	Guest	Visitor of the site with only basic access
A6	Customer Service Operator	A representative of TEAM5OAGS that supports the customer through chat.

A7	Worker	Manages work orders of frames

Table of Use Cases:

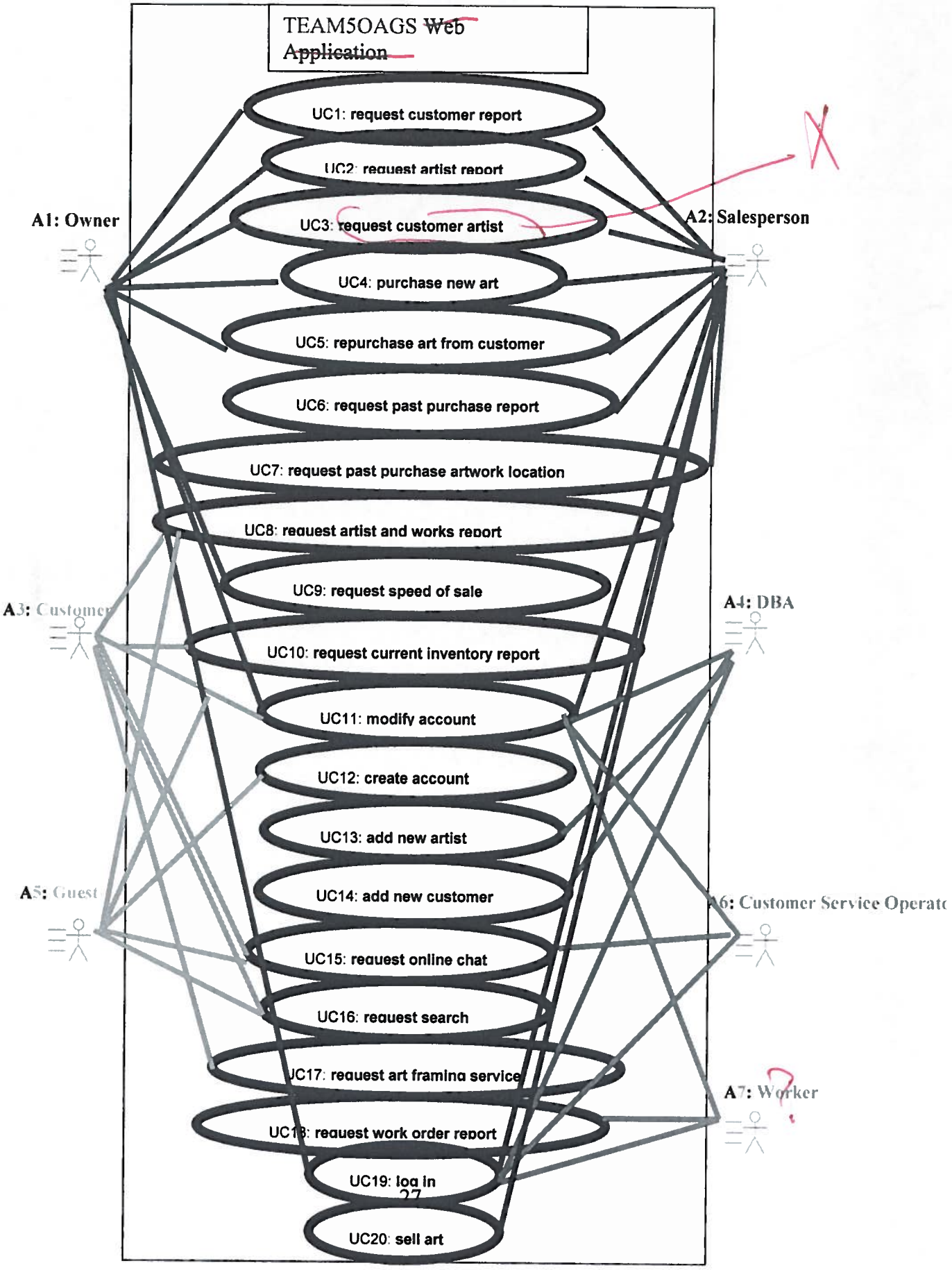
Use Case #	Use Case Name	Brief Semantics	Line
UC1	request customer report	Requests information of customer such as : name, address etc.	67
UC2	request Artist report	Requests artist information	68
UC3	request customer artist preferences report	Requests a report of a customer's preference to a particular artist	70
UC4	purchase new art	Purchase new artwork to sell.	73
UC5	repurchase art from customer	Repurchase art that was sold to a customer.	75
UC6	request past purchase report	Reqests a past purchase report of a customer.	107
UC7	request past purchase artwork location report	Requests a report that has the location of artwork that was purchased.	109
UC8	request artist and works report	Requests a report that has artists and artwork that has been with TEAM5OAGS	113
UC9	request speed of sale report	Requests a report that shows how fast an artworks sells.	114
UC10	request current inventory report	Requests a report of current inventory	117-118
UC11	modify account	Allows user to chance information such as username,	120

methods

		passwords, etc.	
UC12	***create account***	Create an account for a user	121
UC13	***add new artist***	Add new artist in the database	123-124
UC14	***add new customer***	Add new customer	123-124
UC15	***request online chat***	Implements online chat with a customer service rep.	28
UC16	***request search***	Search inventory or testimonial	34
UC17	***request art framing service***	Creates a work order for framing services	44
UC18	***request work order report***	Requests a workorder report	58
UC19	***log in***	Log into an user account	119
UC20	***Sell art***	Sell artwork to customer	40

"C" compilable?!

Case Diagram



Procedure :

UML Use Case Descriptions

Use Case	UC 1 – Request Customers Report
Summary	Customer report is requested from the database
Triggering Event	Request for customer records
Brief Description	Actor requests report and the following information is represented from database • Last and First names • Addresses(street,city,state,zip,country) • Phone numbers(area code and phone number) • Email addresses
Actors	Owner Salesperson
Precondtions	Must be logged in as owner or salesperson
Postconditions	View of request customers report

Use Case	UC 2 – Request Artists Report
Summary	Artists report is requested from the database
Triggering Event	Wanting to look at Artist records
Brief Description	Actor requests report and the following information is represented from database • Last and first names • Nationality
Actors	Owner Salesperson
Precondtions	The Artist Records must exist in database
Postconditions	Artist report view is returned

Use Case	UC 3 – Request customer artist preferences report
Summary	Customer report is requested from the database based on previous bought artwork
Triggering Event	Request to look at customer artist preferences
Brief Description	Actor inputs the customer ID and the customers art preferences are listed
Actors	Owner Salesperson
Precondtions	Must be logged in as owner or salesperson
Postconditions	View or customer artist preference report is retuned

X not matching

Use Case	UC 4 – Purchase New Art
Summary	A new piece of art is purchased and added to the inventory
Triggering Event	New art is acquired
Brief Description	The following information is input when buying new art. • Data about the artist • Nature of the work • Acquisition date • Acquisition price All input into database
Actors	Owner Salesperson
Precondtions	Must be logged in as owner or salesperson
Postconditions	Purchase new art view is returned

Use Case	UC 5 – Repurchase Art From Customer
Summary	A piece of art is repurchased and added back to the inventory
Triggering Event	New art is acquired
Brief Description	The following information is input when buying back the art. • Data about the artist • Nature of the work • Acquisition date • Acquisition price All input into database
Actors	Owner Salesperson
Precondtions	Must be logged in as owner or salesperson
Postconditions	Repurchased view is returned

Use Case	UC 6 – Request Past Purchase Report
Summary	Report to examine past purchase data so that actor can devote more time to the most active buyers
Triggering Event	Trying to sell more art
Brief Description	A report displaying the past purchases
Actors	Salesperson
Precondtions	Must be logged in as salesperson
Postconditions	Past purchase view is returned

Use Case	UC 7 – Request Past Purchase Artwork Location Report
Summary	To identify the location of artworks they have sold in the past
Triggering Event	Trying to sell more art
Brief Description	A report displaying the location of past purchased artwork
Actors	Salesperson
Precondtions	Must be logged in as salesperson
Postconditions	Past purchase Artwork Location report view is returned

Use Case	UC 8 – Request Artist and Works Report
Summary	To provide a list of artists and works that have appeared in the gallery
Triggering Event	Trying to sell more art
Brief Description	A report displaying each artist and the works he/she has done.
Actors	Customer Guest
Precondtions	Must be logged in as customer or guest
Postconditions	Artist and works view is returned

Use Case	UC 9 – Request Speed of Sale Report
Summary	To determine how fast an artist's work sells and at what sales margin
Triggering Event	Request is made to site
Brief Description	Given and artists id a report is given based on what has sold and how much profit
Actors	Owner
Precondtions	Must be logged in as owner
Postconditions	Speed of sale report view is returned

Use Case	UC 10 – Request Current Inventory Report
Summary	Display current inventory on a web page
Triggering Event	Trying to sell more art
Brief Description	Display current inventory on webpage
Actors	Customer
Precondtions	Must be logged in as customer
Postconditions	Current instance of all inventory is added to inventory view and returned to user

Use Case	UC 11 – Modify Account
Summary	All actors have accounts and have the ability to modify their email or password
Triggering Event	Need to change info
Brief Description	Modify view is retuned to change password, email or both.
Actors	Owner SalesPerson Customer
Precondtions	Account info must exist for Actor and Must be logged in
Postconditions	Account info in modified

Use Case	UC 12 – Create Account
Summary	Forms to fill out to assign account information
Triggering Event	New Account information is sent to DBA's view to add new customer or not
Brief Description	All relevant information about account holder in assigned
Actors	Guest
Precondtions	Must not be logged in
Postconditions	Registration view is returned

Use Case	UC 13 – Add New Customer
Summary	New customer information is added to the database
Triggering Event	A new customer is acquired
Brief Description	Customer information is entered into the database • Last and First names • Addresses(street,city,state,zip,country) • Phone numbers(area code and phone number) Email addresses
Actors	DBA
Precondtions	Guest must have already used create an account & Must be logged in as DBA
Postconditions	New customer is added to the database

Use Case	UC 14 – Add New Artist
Summary	New Artist information is added to the database
Triggering Event	A new Artist is acquired

Brief Description	Customer information is entered into the database • Last and first names • Nationality
Actors	DBA
Precondtions	Customer database must exist & Must be logged in as DBA
Postconditions	New customer is added to the database

Use Case	UC 15 – Request Online Chat
Summary	Customer or Guest chats with customer service operator
Triggering Event	Guest or customer request to chat with operators
Brief Description	• Customer service operator chats with guest or customer to answer any questions they might have
Actors	Customer or Guest
Precondtions	Customer database must exist and must be logged in as DBA
Postconditions	Customer or Guest connects with first available customer servie operator

Use Case	UC 16 – Request Search
Summary	Search keys are entered into the search text box and user requests search (Integrated Custom Google Search)
Triggering Event	Customer or guest requests to search for keys in website
Brief Description	• Customer or guest can request a search for particular keys on the website which returns all relevant pages.
Actors	Customer or Guest
Precondtions	Must be logged in as customer or guest
Postconditions	Search results page is returned

Use Case	UC 17 – Request Art Frame Service
Summary	Customer orders frame from the frame stock
Triggering Event	
Brief Description	Customer information is entered into the database • Last and first names • Nationality
Actors	Customer
Precondtions	Customer must be on frame order page and have appropriate input data filled
Postconditions	An order for frame stock is placed with measurements and added to the work order list

Use Case	UC 18 – Request Work Order List
Summary	Worker sees orders that need to be completed
Triggering Event	Worker logged in
Brief Description	• Worker completes current work orders in work order list such as building frames for art orders, hanging art, and shipping new orders.
Actors	Worker
Precondtions	Worker must be logged in
Postconditions	After worker checks that an order is complete it is then removed from the work order list

Use Case	UC 19 – Log in
Summary	Guest logged into website and verifies account
Triggering Event	Username and password is verified
Brief Description	Guest logged into website, username and password is first verified then privileges and home view is returned based on account type
Actors	Guest
Precondtions	Guest must request to log in
Postconditions	If verified account's type view is returned else incorrect password view is returned with chance to register or recovery user name or password

Use Case	UC 20– sell art
Summary	Salesperson uses sell art to process an order for a customer
Triggering Event	Salesperson process order
Brief Description	Salesperson fills out an order form for a customer then hits the process order button.
Actors	Salesperson
Precondtions	Salesperson must be logged in and order form filled out
Postconditions	Order is placed in the system and additional information is sent to work order list

TEAM SOAGS

70

100 points

Web Site Design

Berkowitz & Yash not assigned work.

Master Page ✓ +40 points

Please print single sided!

TEAM SOAGS for Web Site +20 points

-10

Home Page

About Us Page

Contact Us Page

Testimonials Page X -10

+10 points each

Details -10

Assigned but did not do the work -100 points
(no log entry in svn or change history 5-4 5-5)

Typos! -10 points

Master Page is not reused? -10 points

Please print single sided

(70)

TEAM5OAGS

Web Site Design

Version 2.0

Signature Block

COSC 4351	Name	Signature	Date
SE Team Leader	Ryan Lee		3/4/13
SQA	Christopher Cruz		3/4/13
SQA	Muhammad Naviwala		03/04/13

DOCUMENT CONTROL

CHANGE HISTORY

TLs entries (assigned work and due dates) before releasing to the team

Revision	Name	Due Date	Description
1.1	Gabriel Ohlson	02/26/2013	Complete XXX
1.2	Matthew Liang	02/26/2013	Complete YYY
1.3	Joel Loucks	02/26/2013	Complete ZZZ
1.X	Christopher Cruz	02/28/2013	Review Document
1.Y	Muhammad Naviwala	02/28/2013	Review Document
1.Z	Ryan Lee	03/01/2013	Team Leader Final Review and Approval for Purple Deliverables
1.AA	Christopher Cruz	03/04/2013	Looks great. Signing off.

TMs entries when they completed their work

Revision	Name	**Completed Date**	Description
1.A	Gabriel Ohlson	03/01/2013	I completed Wire Frames
1.B	Christopher Cruz	03/02/2012	Reviewed document, looks great.
1.C	Muhammad Naviwala	03/03/2013	Reviewed the document. The wireframes look awesome!
1.D	Muhammad Naviwala	03/04/2013	Looks good. Signing off now.
1.E	Christopher Cruz	03/04/2013	Looks good. Signing off now.
~~1.N~~	~~SQA Name~~	~~XX/XX/XXXX~~	I reviewed Document
~~1.O~~	~~TL Name~~	~~XX/XX/XXXX~~	Final Review and Approval

TLs entry before PURPLE TEAMS DELIVERABLES

Revision	Name	Due Date	Description
2.0	TL Name	03/01/2013	I changed Version to 2.0

DOCUMENT STORAGE

This file is stored in SVN at http://limi.cs.uh.edu/COSC4351/team5/TEAM PROJECT DELIVERABLES/**Web Site Design.doc**.

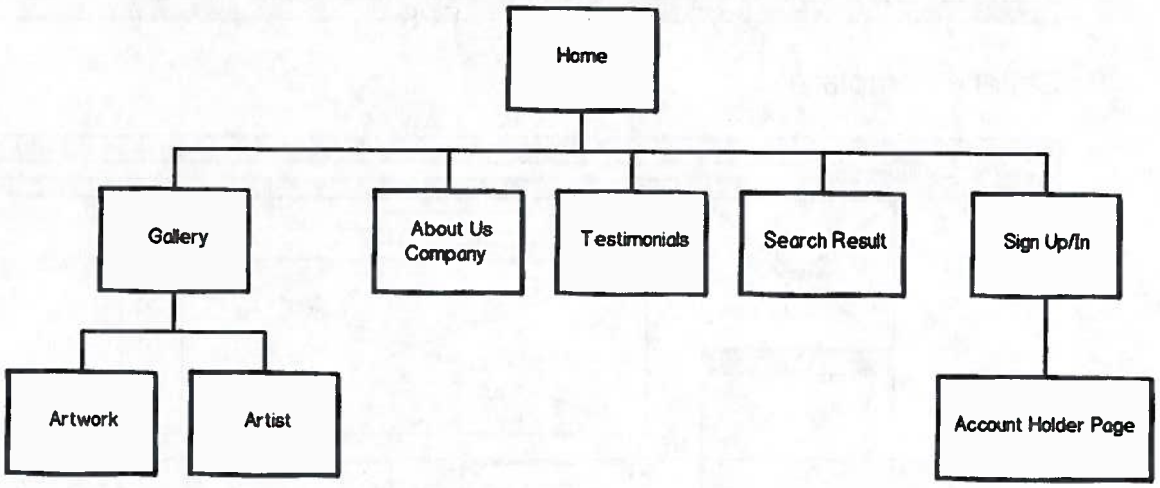
This document includes the hand drawn or tool drawn of your TEAM?OXXS web site design.

You are to deliver AT LEAST ONE page at this time that contains the Master Page and the Home page.

1. Content Outline for Master Page

- **Master Page**
- **Gallery**
- **About Us**
- **Testimonials**
- **Sign Up/In Page**
- **Account Holder Page**

Site Diagram

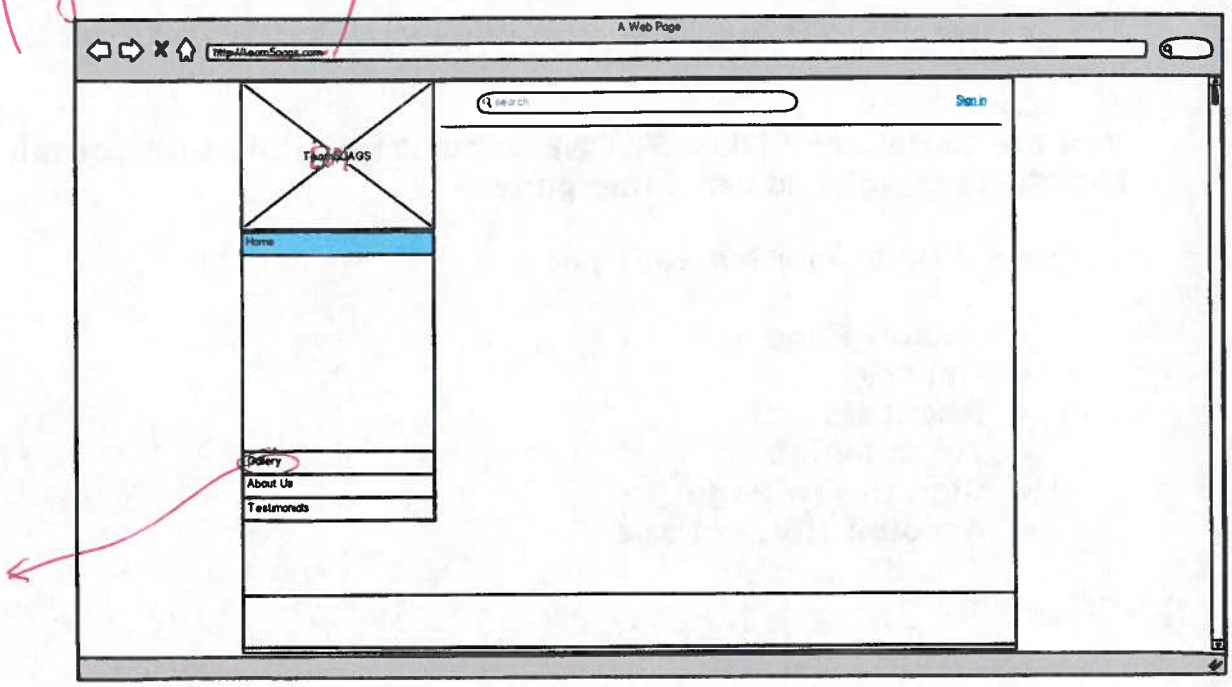


2. Wireframe for Master Page

A. Master Template Page

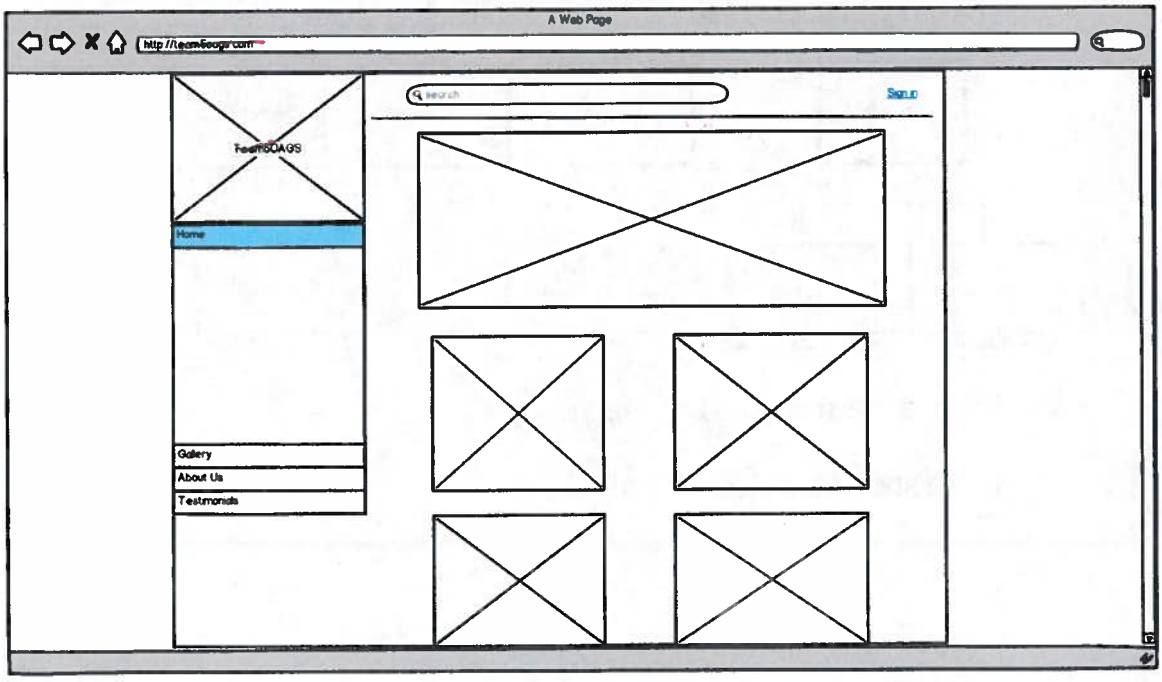
Is this the Master page?

TEAM5OAGS

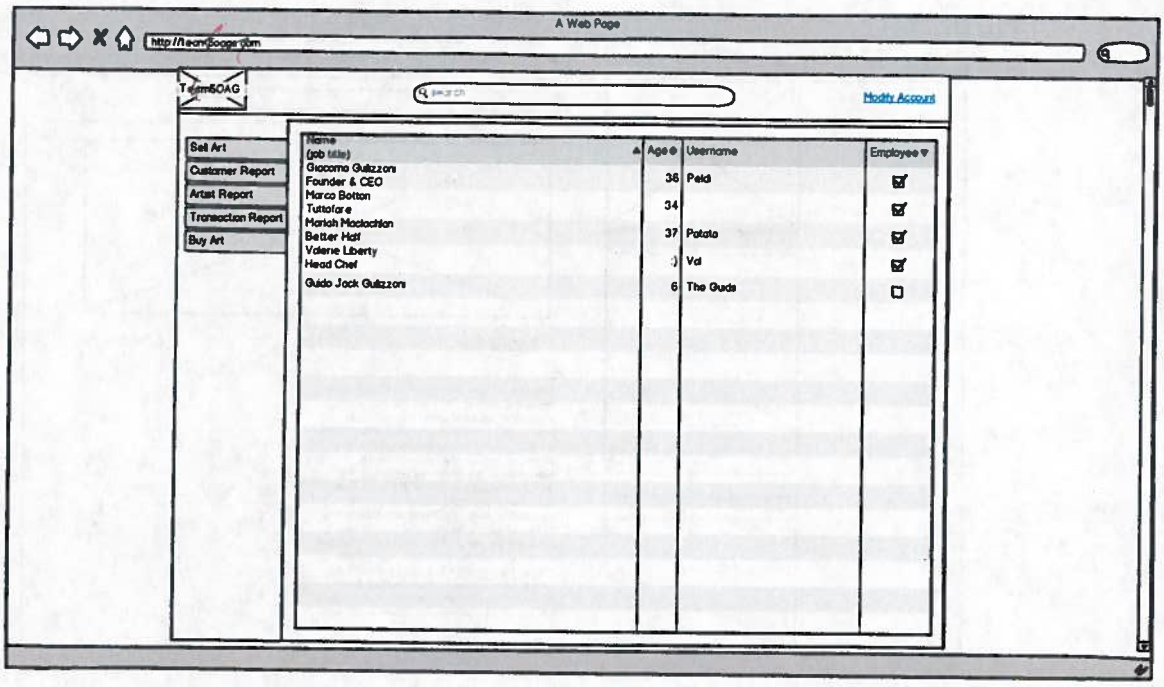


B. Gallery Template

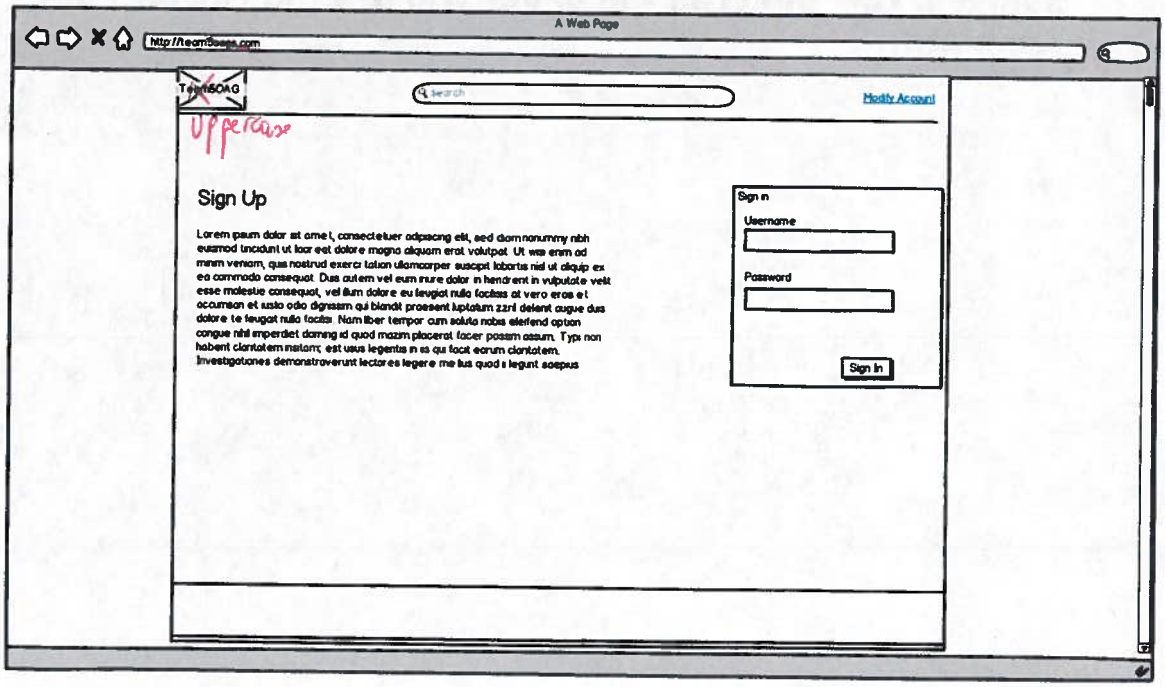
Home



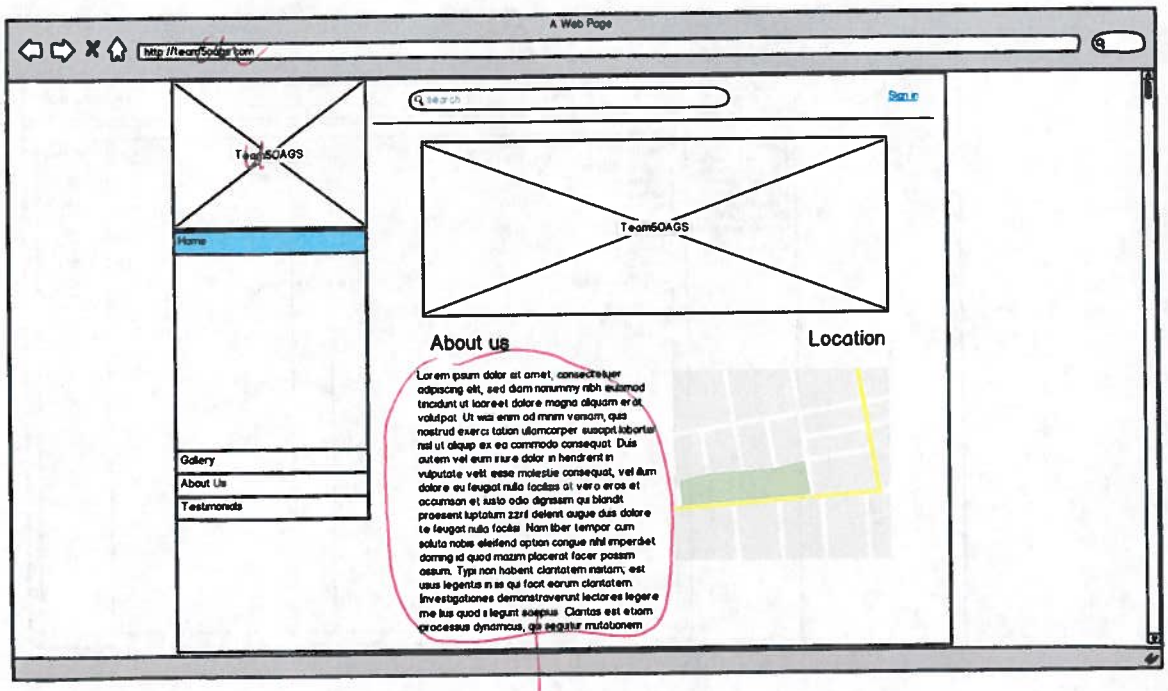
C. Account Holder Template Page



D. Sign Up / Sign In Page



E. About Us



Also, About Us, Contact Us, and Testimonials pages can be added at this time (all the pages will use the Master Page template).

Your Team & Pictures?

Not sure what Galley is?!

BACKGROUND: Site Structure

"Redesigning a website is like remodeling a kitchen - you must figure out what features and capabilities you need and how you will use them before you design your layout, place appliances and plugs, and select tiles, curtains and countertops." -- Web Redesign 2.0

Good web design requires a solid site architecture based on the site's goals and target audience established in the project brief. The deliverables from this phase are:

1. Content Outline
2. Site Diagram
3. Page Description Diagrams
4. Wireframes

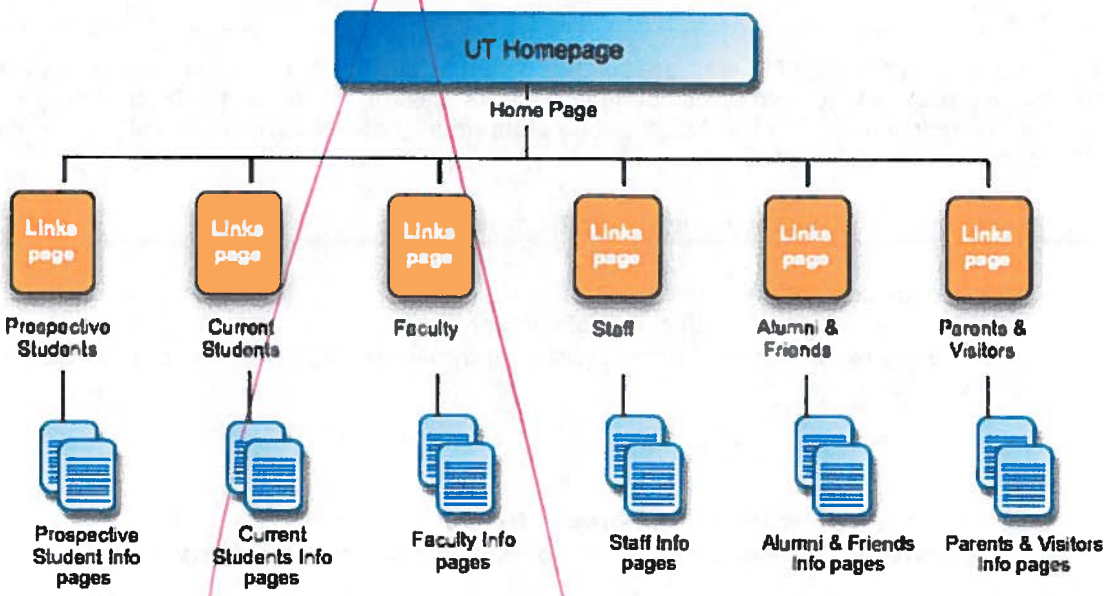
These four deliverables are dependent on each other and need to be completed sequentially.

Content Outline

Working closely with your clients, create a list of all existing content. Brainstorm content that needs to be created for the site. Review the list of content, trimming anything that does not match the goals or audience needs as stated in the project brief. Take time to think about the future and how the site content might need to grow. Make sure you leave room for growth. Next group your content into categories. As you categorize your content, considering getting user feedback through a **card sort**. Once your categories are established, create an outline of your content and review it with your clients for accuracy.

Site Diagram

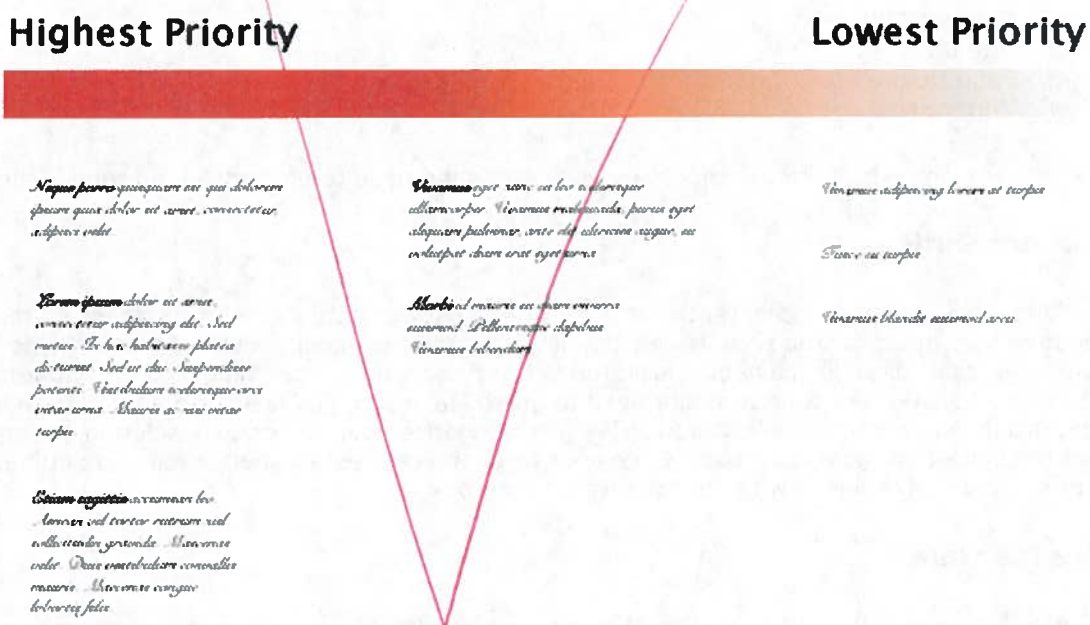
Take your final content outline and create a sitemap or **site diagram**. A site diagram is just a visual representation of your content outline and site structure. You can use Excel, Visio or Omnigraffle to create your site diagram. Refer to the **The Lazy IA's Guide to Making Sitemaps** article at Boxes and Arrows for a step-by-step example.



Page Description Diagrams

Many Information Architects are now recommending the use of Page Description Diagrams (PDD) as a step before wireframes or even as a replacement to wireframes. The focus of a PDD is two-fold:

1. What content belongs on this page
2. What is the priority of each chunk of content



A common layout for the PDD is to use the horizontal access for priority. For example, a PDD for any give page might have three columns. The first column would list the high priority content. The second column would list the medium priority content. The third column would list the lowest priority content.

The advantages of the PDD over the wireframe include:

- clarifies all content for a given page
- clarifies the priority of each chunk of content
- completely removes visual design (color, font, placement) from this stage of the conversation.

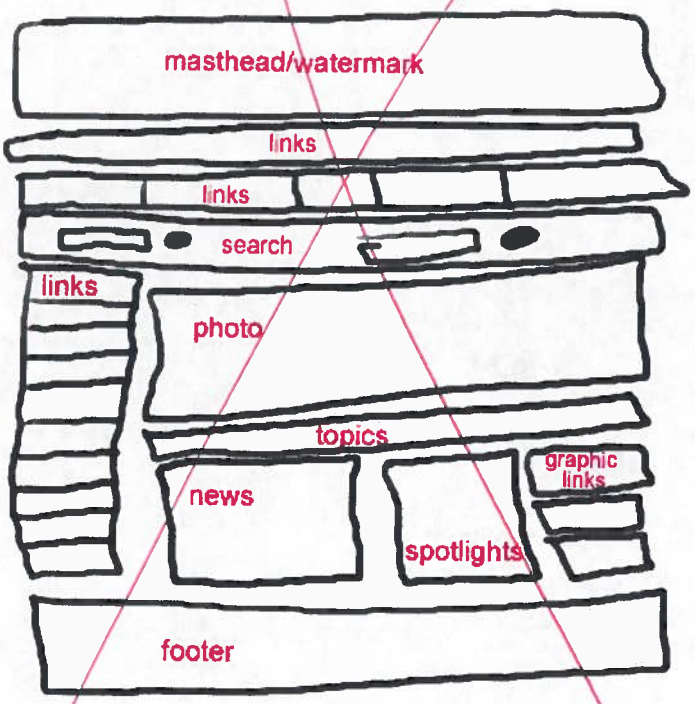
Further reading on Page Description Diagrams:

- **Where the Wireframes Are: Special Deliverable #3**
- **Information Architecture Deliverables: Page Description Diagrams**

Wireframes

A wireframe is a non-graphical layout of a web page. It is a simple drawing of the chunks of information and functionality for each page in your site. You will want to create a wireframe for the home page, each unique second level page and any other significantly different page on the site.

Wireframes include the containers for all the major elements of the page. Elements include navigation, images, content, functional elements (like search) and footer.



Wire framing U.T. homepage

TEAM 50 AG S

45

100 points

Team Project Report (Purple chapters IV, ... VIII)

TM Yash ε 1-8 Berkowitz 1-2 not assigned any work! -100

TDC X X chapter VIII - 20 points

Democratic?

IV (1) Teams "Model"? "CMM level?" X -10 | 20 points

V (2) "SDLC Model?" "CMM level?" X -10 | 20 points

? -5

VI (3) Tools Used Theoretical ε CASE TOOLS | 20 points
UML Models? svn: http:----
Feasibility study? X -5
"Metrics?" X -5

VII (4) Testing | 20 points
"TDD?" X -5
+ SRS Testing "X -10

VIII (5) Developing Motor Page ε Home Page | 20 points
100% match Web Site Design
-10 points "Vague"

Assigned to people? 2 DBAs + 3 TMs!

As per lecture 6, SQAs are to write REVIEW MEETING on SRS with about METRICS! - 20 points

TMs

Please print single sided!

Please print single sided!

TEAM5OAGS

Team Project Report

Version 3.0

Signature Block

COSC 4351	Name	Signature	Date
SE Team Leader	Ryan Lee		
SE SQA	Christopher Cruz		3/4/13
SE SQA	Muhammad Naviwala		
SE Team Leader	Ryan Lee		3/4/13
SE SQA	Christopher Cruz		3/4/13
SE SQA	Muhammad Naviwala		3/4/13
SE Team Leader			
SE SQA			
SE SQA			

DOCUMENT CONTROL

CHANGE HISTORY

TLs entries (assigned work and due dates) before releasing to the team

Revision	Name	Due Date	Description
1.1	Ryan Lee	02/01/2013	Complete Chapter I
1.2	Matthew Liang	02/01/2013	Complete Chapter II
1.3	Christopher Berkowitz	02/01/2013	Complete Chapter III
1.X	Christopher Cruz	02/05/2013	Review Document
1.Y	Muhammad Naviwala	02/05/2013	Review Document
2.1 TL	Ryan Lee 5-1	02/15/2013	Complete Chapters IV, V
2.2 TM1	Matthew Liang 5-4	02/15/2013	Complete Chapter VI
2.3 TM2	Joel Loucks 5-5	02/15/2013	Complete Chapter VII
2.4 TM3	Gabriel Ohlson	02/15/2013	Complete Chapter VIII
2.X	Christopher Cruz	02/17/2013	Review Document
2.Y	Muhammad Naviwala SM's	02/17/2013	Review Document

Roles

TMs entries when they completed their work

Revision	Name	**Completed Date**	Description
1.A	Ryan Lee	01/30/2013	Completed all of Chapter I paragraphs.
1.B	Matthew Liang	01/30/2013	Completed Chapter II
1.C	Chris Berkowitz	01/31/2013	Completed chapter 3 part A. More coming shortly.
1.D	Chris Berkowitz	01/31/2013	Completed chapter 3 part B. More coming shortly.
1.E	Chris Berkowitz	01/31/2013	Completed chapter 3 part C. More coming shortly.
1.F	Muhammad Naviwala	02/01/2013	Reviewed Chapter 1, Chapter 2 and part of Chapter 3. Chapter 1 looks fine. For Chapter 2 part A and B, at some places it is hard to understand what you (Matthew) are trying to say, so I would suggest that you re-read it and make some changes to

			those 2 sections (2A and 2B). For chapter 3, it looks good so far, nicely explained but the tense used is "we" and it looks inconsistent because instead of "we", "the team" is being used in this document. So it has to be consistent. I will be re-reading everything again in the next update.
1.F	Chris Berkowitz	02/01/2013	Added chapter 3 D,E,F. Edits incoming.
1.G	Matthew Liang	02/01/2013	Rewrote Chapter 2 part A
1.H	Matthew Liang	02/01/2013	Add little changes to Chapter 2 part A. Fixed Chapter 2 part B
1.I	Chris Berkowitz	02/01/2013	Made "we" to "Team 5" changes.
1.J	Cruz Christopher	02/02/2013	Read through, change a lot of grammar, punctuation. Reworded sentences, removed sentences that made no sense.
1.K	Muhammad Naviwala	02/02/2013	Read through all the 3 chapters. Made some grammar changes and reworded sentences where necessary.
1.L	Christopher Cruz	02/05/2013	Final Review, Looks great.
1.M	Muhammad Naviwala	02/05/2013	Added the table of contents. Read through the 3 chapters. Looks good. Final review done.
1.N	Ryan Lee	02/05/2013	Team leader final review and approval for Red Deliverables.
2.1	Ryan Lee	02/08/2013	Completed paragraphs for chapters 4 and 5.
2.B	Joel Loucks 5-5	02/10/2013	Started chapter 7 A,B(i,ii). Just getting basic ideas down then will expand later.
2.C	Joel Loucks	02/12/2013	Started chapter 7, accidently started 6 last time, more to come later
2.D	Joel Loucks	02/12/2013	Added more to chapter 7 A & B. Any grammar or written mistakes pointed out would be

			greatly appreciated!
2.E	Christopher Cruz	02/13/2013	Read Chapters 4 & 5, edited grammar
2.F	Christopher Cruz	02/13/2013	Read Chapter 6, edited grammar, sentences, changed some words. Added table of contents. Mo can you fix the leading periods? I couldn't figure it out in word.
2.G	Joel Loucks	02/14/2013	Added more to chapter 7. Any fixes of my grammar is appreciated.
2.H	Christopher Cruz	02//14/2013	Corrected grammar, change some sentences.
2.I	Matthew Liang	02/15/2013	Added part 3 and 4 to chapter 6
2.J	Matthew Liang	02/15/2013	Added B.5, B.6, C, and D to Chapter 6
2.K	Muhammad Naviwala	02/16/2013	Redid the table of contents
2.L	Christopher Cruz	02/18/2013	Checked grammar, and sentence structure. Read through new additions.
2.M	Muhammad Naviwala	02/19/2013	Read through the chapters. Changed some grammar and sentence flow. Chapter 6B-part 7 & 8 remaining. Chapter 7B-part 2, C & D remaining. Chapter 8 not started.
2.N	Matthew Liang	02/27/2013	Added Chapter 6B-part 7 & 8
2. O	Gabriel Ohlson	03/01/2013	Added CH8 Problem Statement.
2.P	Gabriel Ohlson	03/01/2013	Finished Ch8 with wireframes.
2.Q	Muhammad Naviwala	03/02/2013	Read through the newly added content. Made some minor grammar changes. Now only part of Chapter 7 remaining.
2.R	Christopher Cruz	03/02/2013	Read new sections, edited some grammar and reworded some sentences.
2.M	Christopher Cruz	03/04/2013	Looks great. Signing off.
~~2.N~~	~~SQA Name~~	~~XX/XX/XXXX~~	~~I reviewed Document~~

TLs entry before DELIVERABLES

Revision	Name	Due Date	Description
2.0	Ryan Lee	02/05/2013	I changed Version to 2.0

3.0	Ryan Lee	02/18/2013	I changed Version to 3.0

DOCUMENT STORAGE

This file is stored in SVN at http://limi.cs.uh.edu/COSC4351/TEAM5/TEAM PROJECT DELIVERABLES/**Team Project Report.doc**.

Table of Contents:

Chapter VIII ? -5

Report Format and Content

Chapter I: Introduction to TEAM5OAGS

A. Introduction

The purpose of this project is to construct a web application for an online art gallery. The company specializes in the selling of contemporary European and North American fine art such as lithographs, prints, paintings, and photographs. The web application that is to be constructed will allow customers to view and purchase artwork as well as services being offered by the company. The web application will also give merchants and former customers the opportunity to sell their artwork belongings to the company. By using standard software engineering principles and state of the art development tools, the goal of the team is to produce a high quality software product that will not only satisfy client's business needs, but also help improve its business operations.

B. Research Methodology

To determine the feasibility of the project to be undertaken, the team conducted a feasibility study and researched several websites that specialized in the selling of artwork. Examples of websites that were analyzed in the feasibility study were UGallery.com, artmajeur.com, redbubble.com, and 1-sharon-norman.artistwebsites.com. In the study, URLs for the websites and screenshots showing various user interface designs used by each site were compiled into a brief report and described in detail. Also, the web development framework, scripting languages, and servers that each website used were noted in the report. Finally, the team conducted a cost analysis for each website based on the data they have gathered and came up with the cost estimates for each website. Therefore, the team concludes that the project to be undertaken is indeed feasible and can be completed within the time and cost constraints that were estimated in the study.

C. Report Organization

The purpose of this project report is to document and describe the various software engineering principles, tools, and practices that will be utilized in the team project. The report will be divided into ten subsequent chapters where each chapter will focus on a different topic of software engineering. Each topic will be discussed in detail by organizing chapter content with problem statements, body paragraphs, and concluding remarks. The problem statements will present the background information of the topic to be discussed and what issues may need to be addressed. The body paragraphs will elaborate about the issues presented by the problem statement and seek to address them by providing examples and potential solutions. And finally, the concluding remarks section will summarize the topic that was covered by the chapter and may include any recommendations that the team has made for the project.

Chapter II: Expanding the Requirements (Part 1: The Scope of Software Engineering)

A. Problem Statement

History has presented software development as a challenging discipline with rare prospects of success. Although software engineers have been selective on applying theories, methods, and tools that are most appropriate, such practices have presented differences to the way software is developed. This often results in problems such as late delivery, being over budget, and software faults leading to unsuccessful (and unacceptable) outcomes. As TEAM5OAGS is developed, the team will apply the traditional software development practices and address issues from requirements to post-delivery maintenance in a technical and manageable fashion. From an economic perspective, software development comes with a high cost while significant progress has been made. In fact, most costs are associated with the maintenance of the product. For this reason, the team will be tasked with developing a product under a cost-efficient plan for both the company and the client. The product will undergo maintenance throughout the entire life-cycle to address corrections and improvements the software product may undergo.

B. Expanding the Borders

A software product can be modified after delivery to correct faults, enhance performance, or add new functionalities. Maintenance is an essential phase in the software life-cycle because the majority of the costs are associated with the maintenance of the product rather than its development. Also, if maintenance is not done properly, the inability to update software quickly and reliably will result in a lost business opportunity and early retirement for the software product.

Post-Maintenance consists of four different types. First, corrective maintenance involves removing faults or defects while leaving the specifications intact. Next, enhancement maintenance is a software update consisting of the changes to the specifications and the implementation. Third, perfective maintenance involves improvements to the software that add more functionality and efficiency beyond the initial requirements. Lastly, adaptive maintenance involves updating the software product in response to changes in the environment such as new hardware or new government regulations.

The project will undergo corrective maintenance in cases where correcting faults is a necessity. However, the system may also undergo perfective maintenance for improvement oat the request of the client. Overall, promoting a maintenance-friendly product will be the team's priority in terms of effort and spending so that the utility of the product to be developed can be enhanced.

C. Expanding vs. Replacing the System

While it is important to maintain the software, perfective and adaptive maintenances play a big role as they are both essential for allowing the software product to evolve

over time. During perfective maintenance, modifications to the software extend its functionality and improve its performance. This approach is carried out if a client wishes to add new features or modify it to run faster. Adaptive maintenance is performed when changes to the environment the system is running in occur. In such a case, it is imperative to update the system in order for it to fit in the new environment.

TEAM5OAGS will be addressed to handle such changes as the software is being developed. The maintenance will serve to update the system to adapt the new changes as well as surpass its initial requirements through enhancement. Maintenance is vital as it may influence and promote the extent to which this system was originally developed.

D. Concluding Remarks

It is important to recognize that software engineering practices are to be performed skillfully and insightfully. At every step of the software development cycle, they must be conducted in a balanced-budgeted way.

But it is also important to keep maintenance in mind when developing the product because there are bound to be faults in the software. Maintenance will help ensure that these faults are handled in a way that does not affect the integrity of the product. Eventually, the software product will grow to its potential as changes need to be made.

Chapter III: Educating the User ("The Software Process" -textbook & Problems with Software Production – BlackBoard)

A. The Software Process

Developing software is an intricate process that becomes much more manageable with a plan. Team 5 will be using the Unified Process methodology to ensure the software process will be structured, on schedule, and that the final product will perfectly match the product specified. The Unified Process was chosen as the methodology due to its wide acceptance as the standard object-oriented guideline and also because it is easily adaptable to properly fit any individual software needs.

The Unified Process has five key workflows. These workflows are requirements, analysis, design, implementation, and test workflows. Strictly adhering to these workflows will ensure that TEAM5OAGS will be effectively completed. The Unified Process requires modeling throughout the entire Software Development Life Cycle. After receiving the TEAM5OAGS requirements document, Team 5 began modeling via textual analysis. The team has properly analyzed the requirements and created UML use cases, UML class diagrams, and a Model-View Controller structure suitable for this web application. Through constant communication with the client, Team 5 refined the UML models to best capture the entire domain of the application that the client has envisioned.

B. Problems with Software Complexity

Designing software can spiral out of "complexity" control if allowed to do so. To combat this issue Team 5 has taken many steps to reduce the complexity as much as possible.

Each member of Team 5 completed the UML use cases and the UML class diagrams. Once complete, the document was refined from collaborated findings and created one perfect textual analysis that will help guide the development of this application towards the final deliverable. This will not only help lower the complexity of this process but will also keep the team on schedule for the checkpoints throughout the entire project. These models have also helped outline the exact specifications that TEAM5OAGS must have. Using these models and keeping communication constant amongst the team will greatly reduce the software complexity problem, and prevent other project hurdles such as scope creep and going over budget. Combining these processes and through continued use of the Unified Process, Team 5 will keep the software complexity issue at a minimum.

C. Problems with Software Conformity

Problems with software conformity are constantly arising. Parts of a larger software system may be upgraded and find they no longer work within the old product as intended. This is a serious issue that can ruin a project, destroy a necessary process, and even close an entire business down. This project has taken multiple steps to avoid this issue. For this reason, this is why Team 5 has decided to follow the Model-View Controller format. This means that Team 5 will not have to have the product conform to other preexisting products, where integration issues could arise. Team 5 will create all of the parts and the MVC form will help the team maintain this application's modularity for any future system upgrades.

D. Problems with Software Changeability

To address the problems with software changeability Team 5 has decided to implement a few simple principles that will be followed throughout the entire SDLP. Team 5 is going to, as previously mentioned, use the MVC format for TEAM5OAGS. This will greatly reduce many issues regarding ease of changeability in the future. By keeping the code in the MVC form the parts of the system will be much more accessible and therefore changeable if need be. TEAM5OAGS will also be heavily documented and the team will ensure that this documentation takes place throughout the entire process. This will help the users and any future system upgrade with the ability to succeed and understand thoroughly exactly what is going on within the application.

E. Problems with Software Invisibility

The problem with software invisibility lies within the actual software design itself. Team 5 has put considerable effort into creating TEAM5OAGS with the end user in mind. By keeping the application as modular as feasibly possible the application will

allow for extended use of the core functionality while also catering to the ever-changing technology frontier. This affects invisibility because when the application is written there is no emphasis on the end users having any ideas on how the actual processes interact with one another. Instead, they only understand how to use the application for its intended purpose. To combat the issue of software invisibility, Team 5 has made the effort to train (and continue training in the future) the end user as for a larger understanding of the back end that is typically unrealized. This will allow for future modification if need be and a firmer understanding of why the program acts the way it does. Team 5 has also documented every action taken towards the completion of TEAM5OAGS so that any third parties who may need to modify any parts of this application will be able to understand the parts separately and as a whole. This will keep future maintenance and upgrade costs at a much lower rate than the typical software products being produced are costing.

F. Concluding Remarks

The software process is a tedious process that leaves room for many problems throughout the entire cycle. Team 5 has taken an exuberant amount of precautions to deliver TEAM5OAGS without errors or flaws. With repeated communication with the client, Team 5 has gotten to understand exactly how TEAM5OAGS needs to function. Using the models and taking the necessary steps to refine the design and perfect the Model-View Controller methodology, Team 5 has hammered out a perfect model to follow during the coming design and implementations phases. TEAM5OAGS will not only work as the premier art gallery web application it is being designed for, but will also save the client on any future endeavors that the client may have in respect to the application.

Chapter IV: Choosing the Right Team ("Teams" - textbook)

A. Problem Statement

Software products are often too large to be developed by one person within the given time constraints. Although the quality of a software product may be higher if it is developed by one professional, the amount of time it would take to accomplish the task would most likely be unfeasible. In addition, Miller's Law states that the human brain is only capable of processing several chunks of information at a time. Therefore, large software products must be developed by a team of software professionals.

B. Assembling the Team

There are several types of teams that software professionals can form. First, this report will focus on the two classical approaches to forming teams and discuss the nature of such teams as well as their advantages and disadvantages.

i. The Democratic Approach

The democratic team organization was first described in 1971 by Weinberg. He pointed out that the basic concept underlying the democratic team was egoless programming. Egoless programming describes how programmers are supposed to be neutral to the code they write and not see it as an extension of themselves. For instance, a programmer who is passionate about his or her code would be more reluctant to see something wrong with it in the event a bug is uncovered. This form of attachment to the code that programmers write increases the likelihood that unnoticed bugs will be introduced into the product and thereby decreasing the quality of the software product.

The democratic team approach is to have each member perceive the modules that are developed as part of a team rather than belonging to any one individual. Members are encouraged to find faults in their code and accept that the presence of faults does not imply anything bad about them. Rather, the presence of faults in code is considered a normal event that is to be expected and reviewers should not ridicule programmers who actively seek to find faults in their code.

A final aspect of democratic teams is that all members work for a common cause with no "leaders"; in other words, each member works for what is best for the team. In most workplace scenarios, the promotion of a programmer to a management position often results in his or her fellow programmers striving harder to attain the higher level at the next round of promotions. This is a case where team members are not necessarily concentrated on the best interests of the team; instead they are focused on attaining higher levels of position. In a democratic team, no team member works for the sole intention of attaining higher positions in the company

hierarchy. Instead, members work together as a single unit to accomplish the software projects they are responsible for.

Overall, the benefits of democratic teams lead to the production of higher quality software products. Since the attitude towards the finding of faults hidden in the code is positive, each member is highly motivated to debug their codes with high scrutiny. Unfortunately, the democratic team approach is not without its shortcomings. It is impractical in environments where managers and programmers are reluctant to accept egoless programming. It is also impractical in environments where experienced professionals will be working with inexperienced programmers who appraise their code. Although the democratic team approach leads to higher rates of productivity, its impracticality in environments where competition is present among professionals makes it hard to implement.

ii. The Chief Programmer Approach

The chief programmer approach was put forward by Brooks in 1975 when he analogized the chief programmer team to a team of surgeons who were operating under the direction of the chief surgeon. However the chief programmer team is not just limited to the expertise of the chief programmer; other experts in different fields of practice may be introduced if necessary.

There are two key aspects of the chief programmer team. The first is a concept called *specialization*. Specialization refers to a situation where each member of the team carries out only those tasks for which he or she has been trained. For instance, the DBA member of a team will be solely responsible for handling the program's interactions with the database rather than the programming logic of a graphical user interface. The second aspect is hierarchy. Here, the chief programmer directs the actions of all other programmers and is responsible for every aspect of the operation.

The chief programmer is considered to be both a highly successful manager and a highly skilled programmer who is responsible for the architectural design and any critical sections of the code. Usually the chief programmer would be assisted by a backup programmer of similar (but less) expertise. In the event the chief programmer had to take a leave of absence there would be someone familiar with the project to replace them. The other programmers on the team work on the detailed design and the coding under the direction of the chief programmer. In addition, a programming secretary (also called a librarian) is responsible for keeping up with the documentation of the entire project, which often included a copious amount of paperwork. No lines of communication existed between programmers since all interfacing issues were often handled by

the chief programmer. Finally, the chief programmer reviewed the work of the other team members since the chief programmer was responsible for every line of code written.

The chief programmer approach, though it had its shining moment in 1971 with the success of an IBM project associated with the New York Times, is for the most part impractical. This is because it is extremely difficult to find a professional who is both a highly skilled manager and a skilled programmer. Also, finding similarly skilled backup programmers in the even the chief programmer leaves is even more difficult. And finally, the team approach is impractical for projects that will require more than twenty professionals to develop since the amount of interfacing required of the chief programmer will most likely be burdensome.

iii. The Modern Approach: Team Leader and Team Manager

Professionals who are highly skilled in both managerial and programming duties are hard to find in copious numbers to make chief programmer teams practical. The chief programmer is personally responsible for every line of code written and therefore must be present at code reviews. In addition, as a manager, the chief programmer is responsible for the evaluation of team members, which makes the presence of the chief programmer at a code review inadvisable. After all, senior members of a team resent having their works being appraised by less experienced members.

The solution to the problem of the classical chief programmer team approach is to replace the role of the chief programmer with two professionals: a team leader in charge of the technical aspects of the team's activities and a team manager responsible for the managerial duties. The responsibilities of each leader is within the scope of his or her area of expertise. The team leader is responsible for the technical management while the team manager is responsible for duties such as budgetary, legal, and performance evaluation issues. The team leader will be a natural participant in all code reviews while the team manager will be absent since performance appraisal is solely the duty of the team manager.

If a project is large enough, there may be multiple team leaders introduced. Each team leader will be responsible for the technical management of a number of programmers while at the same time be answerable to a single project leader. The nontechnical side is similarly organized. Implementation of the product as a whole is under the direction of the project leader. If the project being undertaken requires a synergistic effort of the team, then channels of communication between each member of the team can be introduced with communication channels from senior

members pointing downward to their subordinates. This allows for a more decentralized system of decision making.

iv. Additional Team Approaches

Alternative approaches to team formation exist with the more recent synchronize-and-stabilize, agile, and open-source programming teams. A well-known practitioner of the synchronize-and-stabilize team is Microsoft. Since the size of the products Microsoft develops is often large (with products containing more than 30 million lines of code being common), the team approach is largely a consequence of the way teams are organized.

The synchronize-and-stabilize model is described by the following scenario: the three or four builds of a synchronize-and-stabilize product are constructed by a number of small parallel teams being led by a manager. The teams consist of several developers working with several testers who work one-to-one with the developers. The teams are provided with the specifications of the product and the members are given the freedom to design and implement their portions of tasks as they see fit. The partially completed components at the end of the day are then put together, tested, and debugged. In this environment, individual creativity and autonomy are encouraged and the components often work together.

The primary strength of the synchronize-and-stabilize model is that creativity and innovation are encouraged, a characteristic of the democratic team. In addition, the daily synchronization step ensures that the developers are working towards a common goal without requiring the communication and coordination of a chief programmer team. While this team approach has been highly successful for Microsoft, it is unlikely to be successfully implemented outside the company since it requires a high concentration of talented managers and software developers.

for TEAM5OAGS

C. Recommendations

The team recommends that the democratic team approach be used due to the small number of members needed to develop the TEAM5OAGS web application. Such a team organization will allow for members to not only actively search for faults within their own code, but also become actively involved in pursuing the team's overall goal. Driven by egoless programming and high fault detection rates, the team believes that the goal of producing a high quality software product TEAM5OAGS can be attained within the time and cost constraints.

CMMX

10

D. Concluding Remarks

Various team formation approaches exist for software professionals, each with their advantages and disadvantages in terms of practicality. Democratic teams, although highly productive in finding faults as well as delivery of high quality

software, are often impractical in workplace environments where competition exists between employees. Chief programmer teams are impractical in light of the shortage of professionals who are highly skilled in both management and programming. Therefore, the most common approach being used in software development is the team manager and team leader approach.

Chapter V: Choosing the Right Model ("Software Life-Cycle Models" - textbook)

A. Problem Statement

A software life-cycle model is a description of the steps that should be performed when building a software product. Since it is almost always easier to perform a series of smaller tasks than one large task, each life-cycle model is broken down into a series of smaller steps called phases. The six classical phases of software development include the following: requirements, analysis, design, implementation, post-delivery maintenance, and retirement. All software life-cycle models incorporate these phases, although the order or level of focus of each phase may vary from model to model. In this chapter, five software life-cycle models will be discussed in detail: waterfall, incremental, extreme programming, synchronize and stabilize, and spiral.

B. Comparing Models

i. The Waterfall Model and the Iterative and Incremental Approach

The waterfall life-cycle model is a classical model that dates back to 1970. It includes all six classical phases of software development and typically a project follows the phases in strict sequential order. The critical aspect of the waterfall model is that no phase is complete until the documentation and testing for the phase being worked on has been completed and reviewed thoroughly by the SQA group. This carries over into modifications; for instance, if the products of an earlier phase have to be changed, then the project must proceed back to the affected phase and the modifications must be performed. In addition, any subsequent phases after the affected phase may need to be updated and checked/tested by the SQA group. Testing is not inherent to any one phase; rather it is performed throughout the entire life-cycle of the product.

The ideal application of the waterfall model would be that the software would be developed in discrete phases (namely the six classical phases.) However such an ideal application is impractical if not realistic in the present world. Instead, application of the waterfall model follows an iterative and incremental approach. Here, the software product (assuming too large to be handled by one professional) is broken down into smaller parts called modules. Each module would then be handled by a team of developers and undergo the six classical phases in sequential order that makes up the waterfall model. Each undergoing of the six phases would take place in an iteration. Iterations take place when one artifact is

produced and is later revised to produce a second version and so on. With each iteration, it is the team's goal to ensure that the new iteration is closer to the desired target (satisfaction of specifications) than its predecessor. When a final version of the module is produced and satisfies its specifications, then the iteration for that module ends.

Incrementation is the process of constructing an artifact (whether it be a module or the entire software product itself) piece by piece. These pieces used to construct the artifact are the results of a series of successive iterations, with the final iteration yielding the completed, satisfactory product. The goal is to have all of the modules that are individually constructed through iterations be put together through many increments to constitute the entire software product. This is accomplished by stepwise refinement, which is the process of concentrating on developing the most critical aspects of the software product first before moving on to the less critical aspects.

The waterfall model's strength lies in the fact that each phase of development is heavily documented, reviewed, and tested by the SQA group. Such a practice would help ensure that the software product being developed is able to meet its targeted specifications. However, the waterfall model has serious weaknesses that make it impractical to be implemented in most professional settings. First, the waterfall model relies on the specifications being static over time. In other words, the specifications must not change over time. The problem associated with changing requirements by the client is often called the *moving target problem.* The waterfall model is notoriously vulnerable to delivering software products that the client initially asked for, but not what they wanted in the end.

In addition, the waterfall model is a rigid model that is inflexible to changing requirements. As requirements are changed over time, certain phases of software development are affected and as a result, the module being worked on or even the entire project must come to a halt. The developers must suspend their current work and go all the way back to the affected phase to make corrections and updates in light of the new requirements. In addition, they must also perform updates for the subsequent phases that are affected as a result of the changes. This process has the unintended effect of delaying the project, which could cause the developers to miss their targeted release date. Such a release delay might not only decrease the level of confidence of the client organizations, but also open up potential situations for litigation.

Overall, the current application of the waterfall life-cycle model follows an iterative and incremental approach, although its true and idealistic

application would not. Some authors go as far as saying that the waterfall and iterative and incremental models are one and the same. This is only true because the waterfall model is applied in an iterative and incremental approach. Although the waterfall model has its strengths in delivering a software product that meets its client's needs, its strength relies on the condition that the requirements remain static over time. Such a condition is highly unrealistic, especially when the activities of a client business could change. In addition, the dilatory effects introduced by such requirement changes make the waterfall model rigid and inflexible for developers. As a result, current application of the waterfall model in the industry is rare.

ii. The Extreme Programming Model

Extreme Programming is an agile development practice that is fairly new to software development. Like the waterfall model, extreme programming has its application based on the iterative and incremental approach.

The first step that the software development team takes is determining the various features the client would like the software product to support. These features are called user stories. For each feature, the team then informs the client the feasibility of the feature, how long it will take to implement it, and how much it will cost. This first step corresponds to the classical requirements and analysis workflows of software development. Based on cost-benefit studies, the client then selects the features to be included in each successive build. The proposed build is then broken down into smaller pieces called tasks. Each task has its test cases drawn up by a programmer; this is called test-driven development. Then two programmers work together on one workstation in implementing the task and ensuring that all the test cases work correctly. This is called pair-programming, an inherent feature in agile development. Each programmer takes turns in coding while the other checks the code. Overall, the implementation of tasks would be done in parallel to ensure that integration is continuous.

Another feature inherent in extreme programming is refactoring. Refactoring is the process of changing the internal structure of a code artifact without altering its external behavior to improve some of the nonfunctional attributes of the software. Since extreme programming (and agile development in general) does not contain a phase dedicated to design, changes in design to an artifact of a product take place by refactoring. When a test case does not run, the code is refactored until a design that is simple and straightforward runs all the test cases correctly.

Implementation takes place much earlier in extreme programming because working software is one of the measurements of progress among agile

developers. In addition, the constant delivery of bits of working software to the client helps elicit feedback that can guide future development efforts. Since the delivery of working software is one of the principles of agile development, one of the goals of agile developers is to set iteration deadlines for implementing tasks. This is called timeboxing and typically the iterations will last approximately three to four weeks. If the task cannot be implemented within the time constraints, then a reduced feature will be delivered. Eventually the iterations release increments to the client will constitute the entire software product to be delivered to the client. In fact, it is possible that given enough increments, the client may be satisfied and further development may not be necessary. As such, extreme programming is often associated with delivering only the features the client needs and nothing more.

iii. The Synchronize and Stabilize Model

The synchronize-and-stabilize life-cycle model is a version of the iterative and incremental approach to software development that is used almost exclusively by Microsoft. Due to the presence of synchronize-and-stabilize teams in Microsoft, the company has been able to implement this model with significant success.

The synchronize-and-stabilize model begins with the developers interviewing numerous clients and extracting a list of features of highest priority. Specification documents are drawn up and work is divided into several builds by order of criticality. Each build is carried out by a number of small teams working in parallel and at the end of the day, all teams synchronize their builds. That is, they put the partially completed components together and test and debug the resulting product. Faults are detected and at the end of the day, the build is frozen (stabilization) and no further changes will be made to the specifications. The repeated synchronization steps help guarantee that the various components constructed will always work together.

iv. The Spiral Model

The Spiral Life-Cycle Model is a model that combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. It allows for incremental releases, or improvements, through each time around the spiral and thus is appropriate for large scale projects. Essentially, the spiral model can be viewed as the waterfall model with each phase followed by risk analysis. Risk analysis is performed to mitigate the risks that are associated with each phase. If the risk cannot be mitigated to an acceptable level, then either the specifications are changed or the project is cancelled.

The spiral model makes use of prototyping for providing certain classes of risk. For example, prototyping can be used to measure whether a conceived product can perform within a certain time constraint. If the prototype cannot achieve the desired results, then it is likely that the product to be developed itself may not be able to as well. This information offered by prototypes help give developers a good idea over the feasibility of mitigating certain risks. There are, however, other types of risks that cannot be modeled by prototypes. For instance, the risk that the software personnel needed to develop a product cannot be hired or that the necessary hardware to construct the product is not delivered on time are risks that are not amenable to prototyping. As such, not every risk analysis that precedes a development phase in the spiral model involves prototyping.

The spiral model of software development has a number of strengths. First, it incorporates software quality as an objective. Second, it is a risk-driven process. No phase is worked on until the proper risk analysis studies have been completed. This check ensures that development efforts on large projects do not go to waste if certain conditions cannot be met, saving potentially huge sums of money. However, the spiral model has restrictions in that it can only be used for large scale projects and that only professionals skilled in risk analysis are hired. There is nothing worse than a team believing that all is well when their project is in reality heading for disaster. In addition, the spiral model, like the waterfall model, assumes that the software product is developed in discrete phases when in reality it is not. Instead, software development follows an iterative and incremental approach.

C. Recommendations for TEAM5OAGS

After reviewing the five software life-cycle models and scrutinizing their characteristics as well as advantages and disadvantages, the team will develop the software product TEAM5OAGS using the iterative and incremental approach. Such an approach is not only widely used, but also practical for developing applications that require stepwise refinement. In addition, the iterative and incremental approach will allow the team to develop code artifacts that not only satisfy their specifications, but also integrate together to form a high quality software product.

There are only 4 !
i
ii
iii
iv
-5

D. Concluding Remarks

Overall, five life-cycle models have been studied in detail with their various features described. The waterfall, extreme programming, spiral, and synchronize-and-stabilize models all share the common characteristic of being applied in the iterative and incremental approach rather than the classical approaches they may prescribe. For instance, the waterfall and spiral models assume that software development undergoes discrete phases when it does not. However, the spiral

model is an ideal model for designing large-scale applications involving high risk while the waterfall model is ideal for projects with static requirements. In addition, the iterative and incremental models such as extreme programming and synchronize-and-stabilize models offer features endemic to them. While various life-cycle models exist in software development, it is important for the team to choose the appropriate model for development as there is no one size fits all solution.

Chapter VI: Analysis and Development Methods ("The Tools of Trade" - textbook)

A. Problem Statement

A CASE tool, which is an acronym for Computer-Aided Software Engineering, helps software engineers in every phase of the development process from conceptualization to implementation.

B. Conventional Engineering Methods

i. Stepwise Refinement

Stepwise refinement is defined *as a means to postpone decisions on details until as late as possible to concentrate on the importation issues*. This is important because Miller's law states that the human brain can only concentrate on approximately seven things at a time, which makes building a software product in one step unreasonable. One of the difficult tasks with stepwise refinement is deciding what are the important issues in each step. A certain task may not be important early on in development, but as each step of development progresses that task may become crucial and therefore would need to be focused on.

ii. Cost Benefit Analysis

Cost Benefit Analysis is a way of determining whether a possible course of action would be profitable by comparing estimated future benefits against projected future costs. This analysis is important in determining if a software product, with the cost of hardware, is economically beneficial to an organization.

When analyzing the benefits of automating a clients business, assumptions must be made in order to determine the initial cost of set up and the long-term savings of having an automated system. This includes what type of product that is developed and how the data is stored.

iii. Software Metrics

Throughout the software process, the software metrics can detect the problems early on so that the potential problems would not become serious later on. Out of many different metrics, product metrics and process metric are two most

important measurements. Product metrics measure the aspects of a product, while process metrics are used by developers as a means of finding problems during their software process. A software company should always use the five essential fundamental metrics: the size on the lines of code, the cost in dollars, the duration in months, and the effort done by person given months and the quality with the number of faults detected. Such metrics are applied by the given workflow and addresses the problems within the software company such as high fault rates during the design workflow.

iv. CASE

Computer-aided software engineering, or CASE, can assist an individual to carry out different tasks with ease but does not automate process for a user. Tasks such as documentation are not typically pleasant to work with, but with computers, it can make a difference in handling long process. CASE can support managers and developers to cope the complexity of software development in managing detailed information for easier access. While it makes development more manageable, computer is just a tool for aiding. The development and maintenance of software still requires human intervention.

v. Software Versions

A software product in maintenance has an old version and a new version. The new version, namely the "revision", is built upon multiple versions since software has been installed with improvements and upgrades. Nevertheless, any old version should not be thrown out because the new revision does not guarantee a better product than the old. Not all will use the latest version of the product due to incompatibility or preference. Also, it is still necessary in case of finding fault in the old version in order to correct the issue. Hence, it is needed to keep a copy of every revision of each artifact.

Unlike revision, variations are created and implemented particularly to replace or coexist with the former variation. Also, variations are needed to create a new version to run on different hardware platforms and operating systems.

vi. Configuration Management

The configuration of the product has the specific version of each artifact from which a given version of the complete product is built. Every artifact consists of codes in three forms: source code (generally implemented in a high-level language like C++ or java), object or compile code (produced by compiling the source code), and executable load images (resulted from complied coded combined with run-time routines).

When an artifact failed on a specific set of test data, programmers attempt to re-create the failure and determine which versions of which variations that produce the error in the product. A version-control tool is one of the solutions that are able to manage multiple versions of artifacts and the product as a whole when a

derivation of every version in the artifact is received. A derivation is a detailed report composed of name of each source code, variation and revision, versions of various compilers and linkers used, the name of the person who constructed the product, and the date and time at which it was constructed.

Another is the configuration-control tool which can automatically manage multiple variations and handle problems caused by the development and maintenance by teams when working on a fault. Configuration-control is usually needed to ensure any new version is correctly embedded into the artifacts during the post-delivery maintenance and the implementation phase. For instance, to search for a fault, a baseline, or configuration of all artifacts in a product, must be set up by the maintenance manager and reserve copies of any artifacts into a programmer's private workspace. The programmer can make any changes in their workspace without impacting other's work. Programmer can freeze the process of the current version of the artifact, so other programmers cannot make any changes to it. Once the changes is made and tested, the new version is installed and the resulting artifact becomes the next baseline version. Changes made to an integrated artifact can affect the product as a whole.

vii. Problem Statement

Though CASE is a tool to assist an individual, the user needs to have the experience in handling one with proper care. There is a saying that "a fool with a tool is still a fool" [Guinan, Cooprider, and Sawyer, 1997]. Therefore, it is vital that users are trained to know their tools well. When training is complete, performance and software quality would improve. Such result justify that CASE Environments should be used at least at maturity level 3.

viii. Other Applicable Engineering Tools

On the other hand, programmers have used other tools such as Email, Spreadsheets, and Word processor to carry out general task. Coding tools provide some good assistance when it comes to providing data dictionary, checking code, and generating report or screen. To assure that the codes written are consistent, readable and errorless, programmers need to have tools like structure editor, pretty printers, online interface checkers, source-level debugger, etc. Structure editor is a text editor that can identify syntax faults in the code without compiling. Pretty printer indents the amount of appropriate spacing to format the code as readable as possible. Online interface checkers is a helping facility which can immediately flag on faults within a method. Developers also use source-level debugger to automatically trace the output to be produced.

C. **Recommendation**

Upon reviewing a number of analytical tools, it is recommended that the Team Project should take advantage of these tools as they give benefits of organizing and scopes to the development. The team will be able to form a high quality software product while performing a stepwise refinement, analyzing the cost-

benefit, measuring the process with software metric, using CASE, building and maintaining multiple versions, and managing configurations.

D. Concluding Remarks

In conclusion, the conventional tools help improve productivity and provide organization for software engineers. These tools are worth investing because they are used as a means of accomplishing a task or purpose while keeping the development effective and efficient in the overall process.

Chapter VII: Testing ("Testing" - textbook)

A. Problem Statement

Testing is an essential part of the software development process. Most software life cycles have a separate testing phase that takes place at the end when the product is being integrated and before post delivery maintenance. This is a major problem for a software product because there could be faults, design flaws and even missing requirements. The terms verification and validation are sometimes used to define this process but are insufficient because they imply that testing should be done at the end of the life cycle, which as stated before, can lead to problems. This is why testing should be carried out during every phase to make sure all the requirements are met, that the design does not have any faults, and that there are no residual faults. There are two types of testing; execution-based testing and non-execution based testing, both of which will be discussed here.

B. Quality Assurance

Faults can happen during every phase of development, and a way to cut down on those faults is to have a dedicated SQA group. The SQA group's job is to ensure that the product is of high quality and establishes the standards to which the product must conform. When developers finish a workflow the SQA's must ensure that the workflow was completed successfully and with high standards. It is not the SQA's responsibility to ensure the product is correct, that is the responsibility of every developer on the team.

i. Non-execution-Based Testing

Non-execution-based testing is where the software is tested without running test cases, reviewing software by reading through it and analyzing software mathematically. Performing reviews of the documentation and requirements with other skilled members of the team called walkthroughs or inspections. The people involved in these walkthroughs would be a representative of the team that drew up the specifications, a manager responsible for the analysis workflow, a client representative, a representative from the team that will be working on the next workflow, and a representative from the SQA group. Since the SQA group is in charge of the quality of work involved in each workflow, the SQA representative should be in charge of the meeting to make sure that faults do not slip through and that the walkthrough is performed correctly since the SQA's have the most to lose if

the product is not up to standards. The whole point of the walkthrough is to point out faults that need to be fixed after the meeting so those in attendance do not waste time. Therefore the walkthrough should be a short meeting and not last longer than 2 hours.

There are two way to conduct the walkthrough, participant driven and document driven. Participant driven is where the attendees present their list of unclear items that they think are incorrect while the representative of the analysis team responds to each query and clarifies what is unclear and if the incorrectness in question are actually faults. Document driven is where the person or team responsible for the documents walk the attendees in the meeting through the document and the reviewers interrupt the review with premeditated questions or questions that come up during the presentation. This type of review is usually more thorough because of the group dynamic of having more eyes on the document and with the presenter vocalizing the document and finding faults that they had not caught before. The primary reason for these walkthroughs is for it to be an interactive process made by the whole review team to find faults and corrections during each phase.

The last way of conducting non-execution based testing is by inspections, and is usually more thorough than walkthroughs by following five steps. The first step is the overview, where the document that is to be inspected is given by one of the individuals responsible for producing that document. At the end of the overview session, the document is distributed to the participants. The second step is in preparation where the participants try to understand the document in detail. Lists of fault types found in recent inspections, with the fault types ranked by frequency, are excellent aids. These lists help team members concentrate on the areas where the most faults have occurred. The third is when the inspection begins; one participant walks through the document with the inspection team, ensuring that every item is covered and that every branch is taken at least once. Then fault-finding commences. As with walkthroughs, the purpose is to find and document the faults, not to correct them. Within one day the leader of the inspection team must produce a written report of the inspection to ensure meticulous follow-through. The fourth step is in the rework, the individual responsible for the document resolves all faults and problems noted in the written report. The last step is the follow-up where the moderator must ensure that every issue raised has been resolved satisfactorily, by either fixing the document or clarifying items incorrectly flagged as faults. All fixes must be check to ensure that no new faults have been introduced. If more than 5 percent of the material inspected has been reworked, then the team must reconvene for the 100 percent re-inspection.

iii.Execution-Based Testing

Execution based testing in defined as the process of inferring certain behavioral properties of a product based, in part, on the results of executing the product in a known environment with selected inputs. While this kind of

testing might seem like the end all it has several troubling implications. "Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for sowing their absence" [Dijkstra, 1972]. Using "selected" inputs and a "known environment" for testing does not reflect the real world use of most programs. Sometimes a program has no control over what type of inputs are being received, like in flight systems, to be able to adequately test all aspects of the product.

So what should be tested? Execution based testing tests five aspects of a product, utility, reliability, robustness, performance, and correctness. Utility is the extent to which a user's needs are met when a correct product is used under conditions permitted by its specifications. This is done by testing the product with inputs that are valid in terms of the specifications and whether the product is easy to use by the client. Reliability is a measure of the frequency and critically of product failure. It is very important to know how often the product fails and how long it takes to fix it during development. Robustness is a function of a number of factors, such as the range of operating conditions, the possibility of unacceptable results with valid input, and the acceptability of effects when the product is given invalid input. A product with a wide range of operating conditions is more robust than a more restrictive product. Performance is the extent to which the product meets its constraints with regard to response time or space requirements. Lastly, correctness is if the product satisfies output specifications, independent of its use of computing resources, when operated under permitted conditions. So if input that satisfies the input specifications is provided and the product is given all the resources it needs, then the product is correct if the output satisfies the output specification.

C. Recommendation for *Team Project*

After reviewing the different approaches to testing, it is recommended that Team5OAGS test during all phases of development. One way will be to have walkthroughs or inspections to test the correctness of our specification documents. We will also used execution based testing to ensure the 5 aspects are met by testing the utility, reliability, robustness, performance and correctness of our product.

E. Concluding Remarks

In conclusion, all aspects of testing are very important to the software development cycle and will be carried out during all phases to ensure our product is of a high quality.

Chapter VIII: Developing ~~*Team Project*~~ Page Master and Home Page ("Requirements Workflow" - textbook)

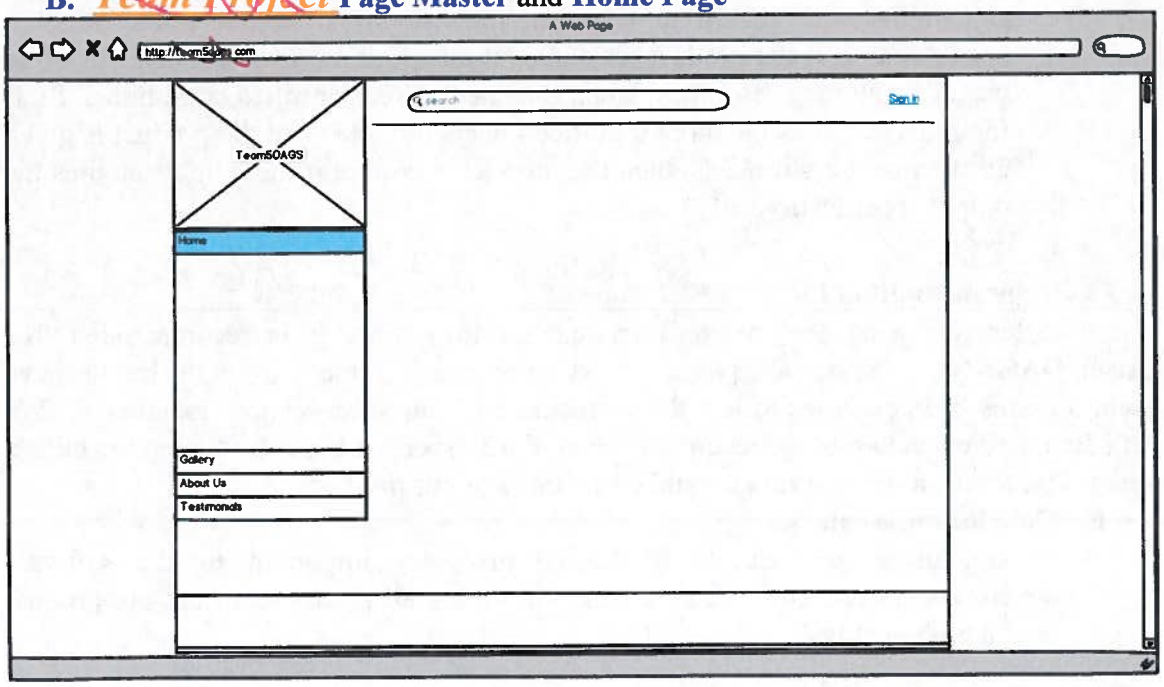
A. Problem Statement

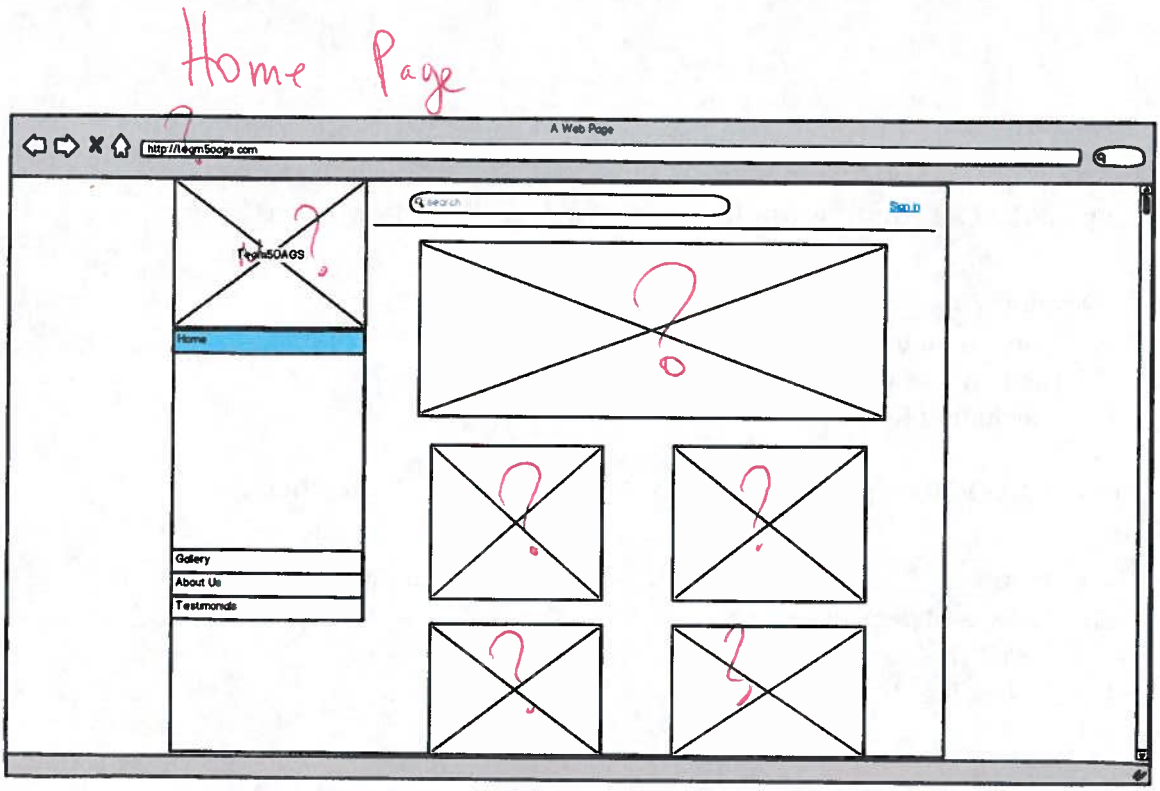
The overall aim of the requirements workflow is to determine the client's needs. Determining what the client needs is often quintessential to developing quality software. The first step to determining this is to understand the application domain or the specific environment in which the product is to operate. Developers must become familiar with the domain before asking the client questions about the domain. This way the developers can ask meaningful questions using nomenclature appropriate for that domain to avoid confusion and misunderstanding. A common misconception is that the developers must determine what the client wants. In reality the client often does not know what they need and the client may have difficulty conveying these ideas to the developers. A client can often ask for the wrong software because software is inherently very complex.

The product must also be functional not only for the developer but also the client. To help alleviate these issues a rapid prototype is constructed which will incorporate key functionality of the target product. Error-checking, file-updating routines and complex computations will not be included. The rapid prototype will reflect what the client needs, such as input screens and reports, but omits functionalities such as file updating and user friendliness. The purpose of rapid prototype is to build it as quickly as possible to provide the client an understanding of the product even if the product crashes every few minutes and or the screen layouts are less than perfect.

TEAMSOAGS

B. Team Project Page Master and Home Page





C. Concluding Remarks

The wireframes will help the client and intended users can examine and experiment with the prototype while the development team takes notes. The developers can change the prototype until both the client and the development team is satisfied that the client's requirements are encapsulated in the rapid prototype.

Chapter IX: OO Analysis Models ("OO Analysis Workflow" - textbook)

(Develop *Team Project* UML OO Analysis Models)

A. Problem Statement
B. The UML Diagrams
C. Concluding Remarks

Chapter X: OO Design Models ("OO Design Workflow" - textbook)

(Develop *Team Project* UML OO Detailed Design models)

A. Problem Statement
B. The UML Diagrams
C. Concluding Remarks

Chapter XI: Deployment Diagram ("More on UML" - textbook)

(Develop *Team Project* UML Package and Deployment models)

A. Problem Statement
B. The UML Diagrams
C. Concluding Remarks