INSE-6270 (Quality Based System Engineering)

RECURSIVE OBJECT MODELING APPLICATION

PROJECT REPORT

Submitted to: Dr. Yong Zeng

Submission Date: April 12, 2007

Project Participants:

Jianqiang Lin, Nick Simmons, Tanvir Khan, Rrima Ima, Yingbo Xu, Zhi Li

TABLE OF CONTENTS

I. First deliverables

I. First deliverables

Quality Planning

1.1 Establish the Project

1.1.1 Mission and Vision Statement

Mission: To design and develop a prototype of the ROM software system.

Vision: To complete the prototype of the ROM system that will automatically translate user requirements from natural languages to graphic representations (ROM diagram).

1.1.2 Establish a Team to do the Planning

See Appendix B – Team Charter

1.1.3 Plan the Execution of the Project

See Appendix D - Gantt Chart

1.2 Identify Customers

External customers: Professor Zeng is the major external stakeholder for this project.
Internal Customers: Quality Control is the customer for all Developer work products.

1.3 Discover Customer Needs

- Customer requires an easy and efficient way to turn natural language requirements into an ROM diagram.
- Goal is to complete a prototype for a trivial input, supporting the basic functionality of transforming natural language into ROM. Prototype should support the following features:
    - Should apply the basic rules for transforming natural language into ROM diagrams.
    - Should be able to analyze a natural language sentence and find the attributes for each word.
    - Find the exact meaning of each word (from an online dictionary)
- The system should be developed using Microsoft .NET, in C#.
- Information should be stored using XML.
- Word definitions should be found using the Google API and online dictionaries.

1.4 Develop the Product

See Appendix A – Meeting Minutes and Appendix D - Gantt Chart

1.5 Develop the Process

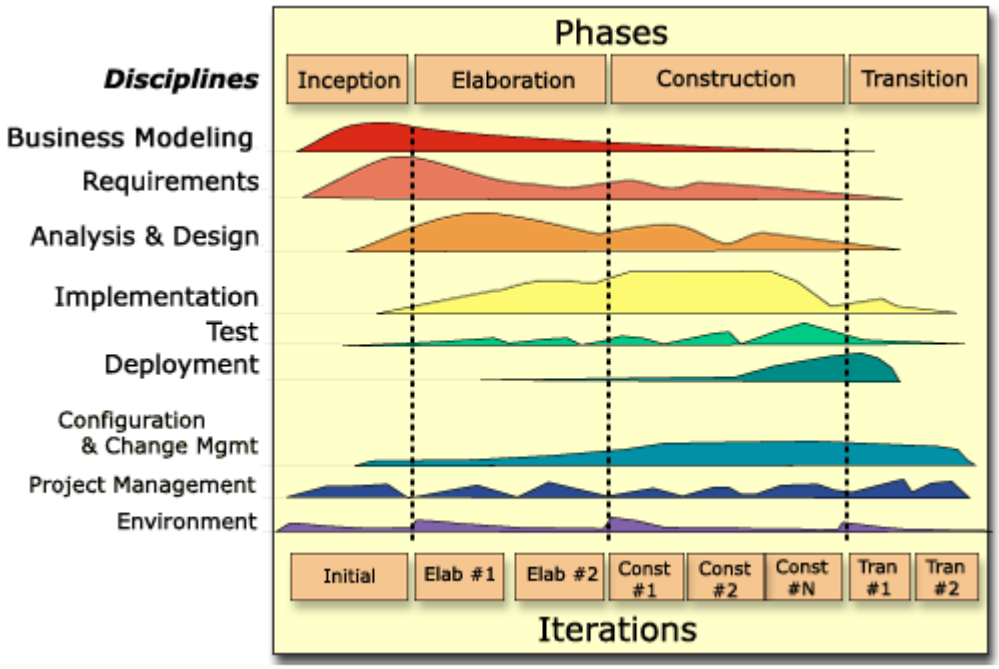
We decided to use a well known process for developing software called the Rational Unified process. It fits well with our requirements and method of working. RUP is based on a set of six key principles for business-driven development:

1. Adapt the process
2. Balance stakeholder priorities
3. Collaborate across teams
4. Demonstrate value iteratively
5. Elevate the level of abstraction
6. Focus continuously on quality

The RUP lifecycle is an implementation of the spiral model. The RUP lifecycle consists of four phases:

- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

Due to the limited time we have in completing the project, we will only be doing the inception and elaboration phases of the RUP model. This is perfect for what the expected final deliverable should be - an initial prototype focused on the key functionalities.



1.6 Develop Process Controls and Transfer to Operations

1.6.1 Identify Controls Needed and Design Feedback Loop

Control needs:

Schedule Control: Meeting Gantt chart, feed back on
Task Assignment: Meeting Gantt chart,
Organization:
Communication:

1.6.2 Feedback Loop:

We should follow the feedback loop diagram as following:

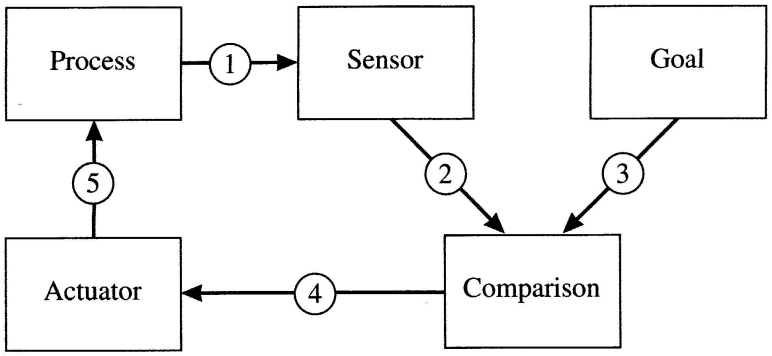
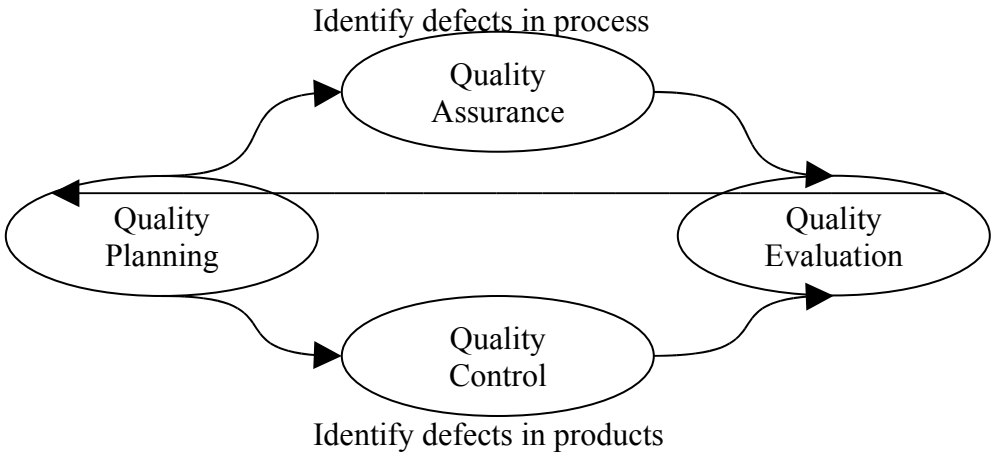


FIGURE 5.1
The feedback loop.

Choose the control subject
Establish measurement
Establish standards of performance: product goals and process goals
Measure actual performance
Compare actual measured performance to standards
Take action on the difference



Quality management activities identify improvements in process and products

1.6.3 Optimize Self Control and Self Inspection

During the planning phase, each task that was assigned to a team member was explained, and the excepted output of that task was defined. The person assigned to each task was the owner of the final product of that task. As such, each team member was responsible for producing the output of that task and ensuring that it meets the standards required. The task owner was expected to perform self-inspections before submitting any preliminary drafts for peer review by the team. Any required corrections must be made and verified before producing the final output for the task.

1.6.4 Team Evaluation

Strengths:
Weaknesses:
Opportunities:
Threats:

View	How			
	Team leader	**Developer**	**Quality Assurance**	**Quality Control/Improve**
Team leader				
Developer				
Quality Assurance				
Quality Control/Improve				

2 Quality Control

2.1 Team Quality

Item	Describe	Goal
Team Communications	Each team member receive the project information	100%
Team Coordination	Everyone conduct project work together	90%
Team Meeting	Everyone present the meeting, every time have log and prepare meeting materials	90%
Team Schedule	Everyone work on the Gantt chart and finish work on time	90%

2.2 Team Approach to Quality

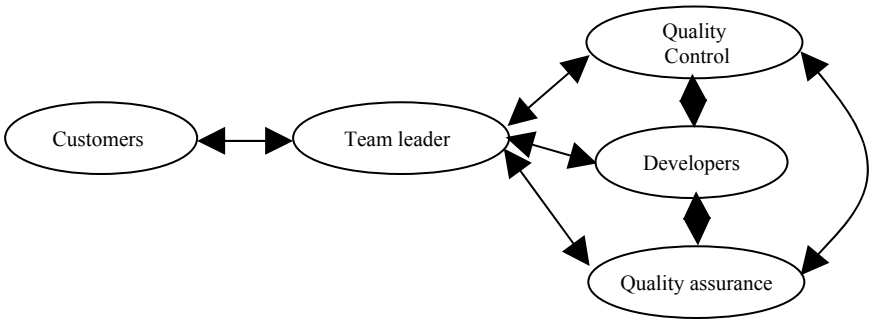
We will use the tools to control team quality.
Communications: hard copy, notice, web publish, email, telephone, Yahoo messenger, MSN messenger and text messenger of mobile phone etc.

Associations: team meeting, team discuss, team communication
Team meeting: meeting log, meeting review, meeting preparing
Team schedule: Gantt chart, Microsoft Project, performance report.

2.3 Quality Control Tools

2.3.1 Communications

The team leader understands that it is very important to know what kinds of information should be distributed to each team members. The communications between teams show as following:



Communication between teams

Information Distribution

After the information is created, it's also important to get project information to the right people at the right time through an efficient way. Most customer management staff tend to read all the information in hard copy while all the programmer in the software development group like to check their email box to get new information.

In our project, we use five ways to keep all the stakeholders informed. When and how they should be used is listed below:

1. Hard copies: This means very important files need to sign on them and keep in the archives. They can be reviewed and photocopied used as a proof when ever necessary.
2. Notice papers: This type of paper used to inform ordinary information always to a part of group or team members. It can always be found at the entrance of the room or building.
3. Web publishes: There are some kinds of information need to update or publish regularly on the web site to inform all members.
4. Emails: which are used in most case, and they are also the most efficient distribution method. Sometimes it's necessary to know if the receiver has got the email or not by adding compulsively reply method.
5. Instant communication: This method includes telephoning, Yahoo messenger, MSN messenger and text messenger of mobile phone etc.

Performance Report

Performance reports describe where the project team stands at a specific point in time. The team members should report the performance report once a week to the project leader, the programmer should report there work status everyday to team leader.

2.3.2 Responsibility Assignment Matrices

See Appendix D - Gantt Chart

2.3.3 Gantt Chart

See Appendix D - Gantt Chart

2.3.4 Quality Control Summary

3 Quality Improvement

3.1 Prove the Need

As the team members found that they were lack of technical expertise needed to complete the project, it was mandatory for them to struggle to gain knowledge on the specific domain. Without the required level of knowledge of some programming languages, it was not possible to develop the software.

The second problem was - the project team had a hard time working together, and it was affecting the project. It had eight people in the group. Some of the team members have strong opinions about how the project should be managed, and others didn't say that much. On the surface, everyone got along okay, but comments on the side led us to believe that there were some conflicts under the surface. The members started thinking after a few weeks of working together -- what should we do to improve our ability to work together as a team?

This later part is something that can sink a project: team effectiveness and cohesion. The team has a group of people, most of whom is a good performer. However, when they get together on the same team, the whole is not as effective and productive as the individual performers.

3.2 Identify Potential Team Improvement Projects

One of the big problems with dysfunctional teams is that much of the trouble lies just under the surface. To identify the problems, we had a group exercise that allows the team to critique itself in a safe environment and then come up with some potential solutions.

One of the team members was the facilitator who helped people open up more. If the leader of the project takes a lead role, the discussion might be more inhibited, especially if the team views the project manager as a part of the problem. So, we avoided the leader not to be the facilitator.

First, participants write on Post-it notes the top five problem areas that they think make the team less effective. This is a brainstorming effort with no right or wrong answers.

The facilitator gathers the cards and posts them on the wall. Problem areas that are related should be grouped together. The team should end up with some general areas, such as lack of leadership, poor planning, and personality conflicts.

Next, the team ranks the general problem areas in terms of importance. Since the original list was built through a brainstorming process, now is the time to try to identify those

areas that are most important. One way to prioritize is to give everyone five sticky dots that they can place by the problem groups they think are the most important. The number of dots by each area will indicate its ranking compared with the other problem areas.

We looked at each problem area individually, starting with the most important. The facilitator asked each person to write down a few ways that the problem could be remedied. All of these potential solutions were constructive and as practical as possible. These solutions are again posted on the wall by the facilitator.

The group discussed the pros and cons of each solution so that everyone understands what is involved. Some solutions were similar and were grouped together. After everyone understood the various solutions, the team again voted on which recommendations make the most sense to implement.

When the recommendations were voted on and accepted, the team became specific in terms of the action plan that will solve the problem. We had a set of activities; people assigned, and end dates. It might make sense to create a section on the project work plan to account for this time.

The team continued with this process until all important problem areas were discussed and solutions put into place. Normally, the project leader was responsible for assigning and following up on the work assignments (after making sure that the member has that particular expertise as this project tasks are mostly not generic).

3.3 Organize Project Team

After having a clear concept of the problems the team was having, we divided the team into two subgroups based on the educational background of the members. One is for development of the software and the other one is for quality issues of the development process and the team itself. Though the team leader was for development, he presided over all the meetings and kept an eye on the milestones. As three of the team members are from computer science background, we had no confusion to assign them for the development. Among the other three, one of the members has two years work experience in testing and quality assurance of the software, so he was responsible for checking the quality of the software based on the set of requirements. It is important to note that assigning the tasks were based on the own preferences of the members, not selected by the team leader.

3.4 Diagnose the Possible Causes of Team Failure

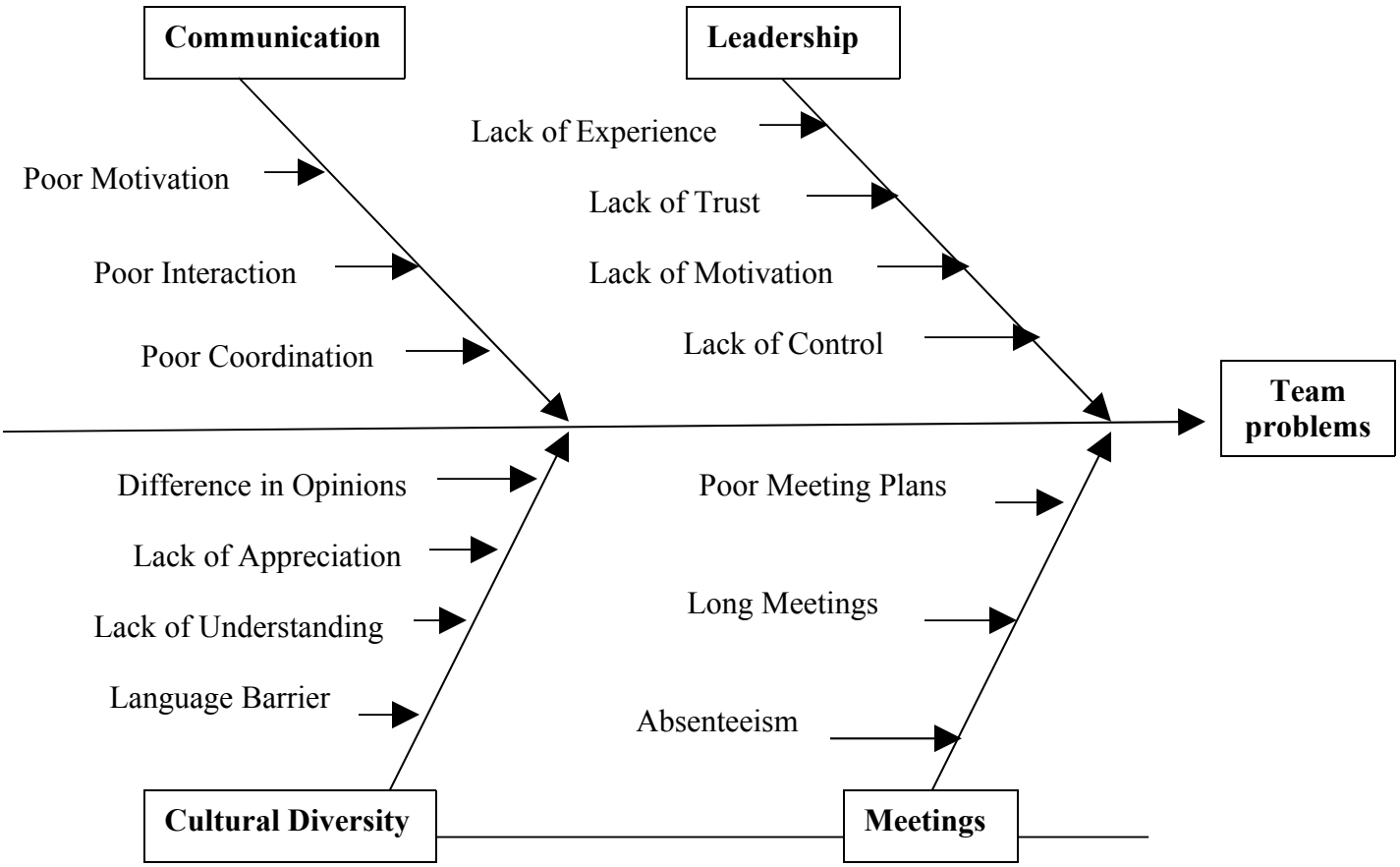
A team is a group of people interacting in order to achieve a common target, which implies a distribution of tasks and convergence of member's efforts to accomplish its goal. However many problems within the team cause the project failure.

Team problems have a negative impact on project success. To reduce the possibility of failure, we identified the factors leading to project failure: Communication, Leadership, Cultural diversity and Meetings.

- Communication is a two-way thing. Good communication means being able to share the information, received, understood and appreciated. However, ineffectual communication is due to poor listening and explaining perceptions, as well as lack on discussing the differences point of views, lack of recommending and negotiating agreement.
- Leadership is a key factor to achieve project goal. Leader must be strong, directive, controlling, confident and give good instructions to his team member as well as some autonomy. Leader has to define the rules for interpersonal relations, to develop the co-operation and to set up responsibility of each member of the team in order to achieve the team's objective. However many leader lead their teams to certain failure due to their incompetence to sustain effective action or create positive change. Besides, rigid leader can be unable to adapt new ideas or new information.
- Cultural diversity can have a positive impact if the team is able to integrate these skills effectively. However this diversity can have a negative impact if each person try to control the others member, do not listen to different opinions or understand others value.
- Meetings provide time to listen, learn, and make plans. However unsuccessful meeting is a waste of time since no clear purpose or outcome results for this meeting.

Cause and Effect Diagram

In order to eliminate the team failure factors and ensure project success we identified the potential causes of each factors.



Proposed Recommendation

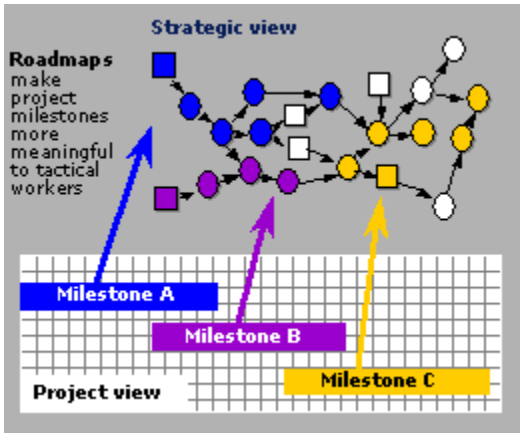
To improve the critical factors affecting project success, we came up with some possible recommendations to be implemented:

- Build team trust.
- Maintain collaborative relationships.
- Preserve a respectful attitude towards the team members.
- Use email and telephone to communicate between team members.
- Cooperate with each other rather than compete.
- Commitment to mission or purpose.
- Agreement on purpose.
- Leader track project progress.
- Leader Keep team member informed and on track

- Leader encourage active involvement
- Leader Delegate work load and Clarify responsibilities
- Team leader establish objective in each meeting
- Establish specific free time for team meeting
- Make a team schedule.
- Distribute minutes of meeting.
- Make sure objectives are met by the end of meeting.

3.5 Provide Remedies and Prove that the Remedies are Effective

As the team was frequently falling behind the schedule, the members decided to have some milestones to make sure that that there is no way we can miss the deadline. As explained earlier, the team members didn't have the expertise in the specific programming languages required for the development. That's why they needed more time to gain the expertise on the domain and decided only to work for development for two months of the project. At this time, they are not responsible for writing up any report or quality issues of the team performance. This was a major change of the Gantt chart or task assignment that we made at the beginning of the project.



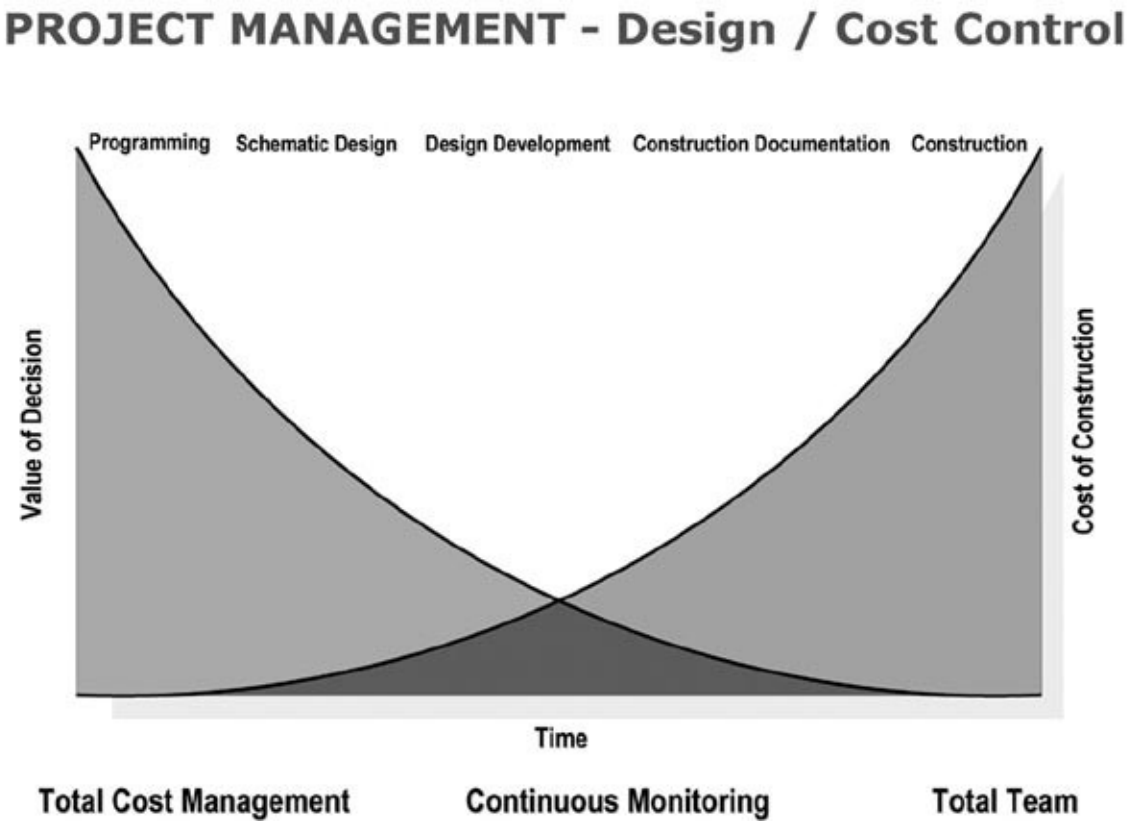
After giving the developers more time for their specific job, they started making progress very quickly. At the weekly meetings, when the members had a discussion about their improvements, they became more confident about meeting the deadline and handing over an advanced version of the prototype.

The benefits the team could see after implementing the solutions were:

- Better role and style fit of project team members
- Faster project team ramp-up
- Increased collaboration
- Increased project team understanding of the importance of their work
- Decreased cycle time for each process

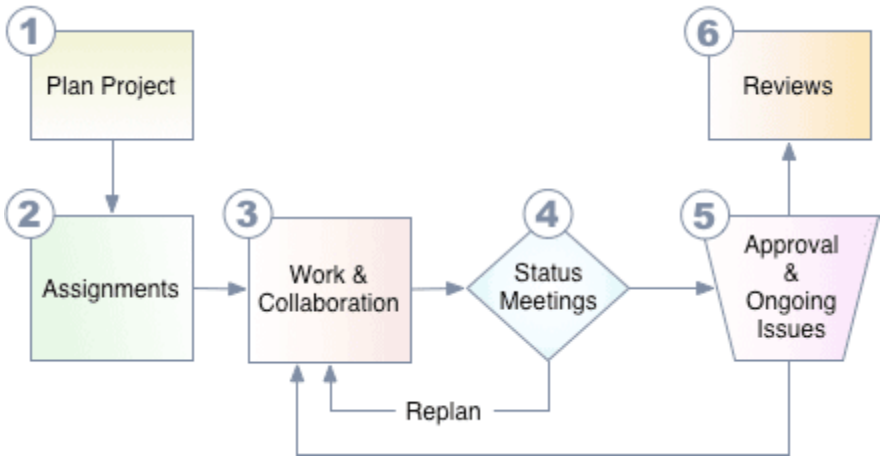
3.6 Deal with Resistance to Change

The team checked the progress of activities against the plan. Review performance regularly and at the stipulated review points, and confirm the validity and relevance of the remainder of the plan. Adjust the plan if necessary in light of performance, changing circumstances, and new information, but remain on track and within the original terms of reference. The team used transparent, pre-agreed measurements when judging performance. We identified, agreed and delegated new actions as appropriate. We planned team review meetings and stuck to the monitoring systems. We analyzed causes and learnt from mistakes. We identified reliable advisors and experts in the team and use them.



3.7 Control to Hold Gains

At every status meeting, whenever the team members found any flaw at planning, they started re-plan and rethink of the issue. The model we started working can be described by the flowchart:



1. **Plan.** Projects are defined by clear objectives, tasks as well as resources to be used, including time. Members of the project plan the flow of tasks as well as the tasks themselves to meet their objectives, given their available resources. Projects typically have a work breakdown structure, tasks that comprise the project, task dependencies, and a planned completion date.

2. **Make Assignments.** The project team makes assignments based on available skills and resources. Contentions for resources are reviewed and resolved in order to select the right people for the project.

3. **Work and Collaborate.** Team members both collaborate and coordinate by regularly posting the status of their own work, participating in discussions with other project members, and providing feedback on tasks done by other team members.

Monitoring mission-critical metrics, such as status updates, elapsed hours and projected progress may be set up to track critical tasks and processes.

4. **Status Meetings.** With all attendees having the same project status information, meetings can quickly and accurately determine which corrective measures should be taken. Projects needing re-planning and mid-course corrections get the attention they need to produce the desired results, as quickly as possible.

5. **Approval and Ongoing Issues.** Tracking ongoing issues makes the team responsive to changes in conditions or specifications. Team members may readily collaborate to create resolutions as well as give approvals to those changes. Regular discussions of issues help identify and resolve issues.

6. **Reviews.** Though the team is not at this stage now, we have plans for the review of the project once this is finished. Project "post-mortems" attended by all members and customer (for this project, the professor of the course) improve planning and processes for all future projects as some of us are planning to continue with this project even after finishing this course. Reviews help to compare estimated time to their actuals along the entire WBS structure.

II. Second deliverables

4 Planning Process

4.1 Mission Statement

The objective of our project is to design and implement the ROM system that automatically translates user requirement from natural languages to graphic representations (ROM diagram). ROM diagram is used to present the relationships between words in a given natural language file. The input of this software is a user natural language document written in English and the output is the corresponding ROM diagram.

4.2 ROM theory

Axiomatic theory of design modeling is a logical tool for representing and reasoning about object structures (Zeng, 2002). It provides a formal approach that allows for the development of design theories following logical steps based on mathematical concepts and axioms. The primitive concepts of universe, object, and relation are used in the axiomatic theory of design modeling. Their definitions can be found from the Random House Webster's Unabridged Dictionary as follows.

[***Definition 1***] The universe is the whole body of things and phenomena observed or postulated.

[***Definition 2***] An object is anything that can be observed or postulated in the universe.

It can be seen from the two definitions above that universe is the whole body of objects.

[***Definition 3***] A relation is an aspect or quality that connects two or more objects as being or belonging or working together or as being of the same kind. Relation can also be a property that holds between an ordered pair of objects.

[***Definition 4***] Structure operation, denoted by ⊕, is defined by the union of an object and the relation of the object to itself.

[***Definition 5***] A product system is the structure of an object (Ω) including both product (S) and its environment (E).

[***Definition 6***] Product environment boundary, denoted by B, is the collection of interactions between a product and its environment.

[***Definition 7***] A design problem can be literally defined as a request to design something that meets a set of descriptions of the request.

Based on these concepts, two axioms are defined in the axiomatic theory of design modeling.

[***Axiom1***] Everything in the universe is an object.

[***Axiom 2***] There are relations between objects.

It can be seen from these two axioms that the characteristics of relations would play a critical role in the axiomatic theory of design modeling. We need to define a group of basic relations to capture the nature of object representation. Two corollaries of the axiomatic theory of design modeling can be used to represent various relations in the universe.

Theorem of Design Problem Structure A design problem is implied in a product system and composed of three parts: the environment in which the designed product is expected to work, the requirements on product structure, and the requirements on performances of the designed product.

According to axiom 1 in the axiomatic theory of design modeling, everything in the universe is an object. Therefore, the basic unit in the ROM is an object represented by a solid box as shown in Error: Reference source not found.

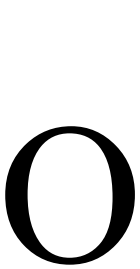


Figure 1 Graphic symbol for object.

A more complex object, such as ⊕O, can be represented by the concept of compound object, which is shown in Error: Reference source not found. A compound object is an object that includes at least two objects in it.

Figure 2 Graphic symbol for a compound object.

According to axiom 2 in the axiomatic theory of design modeling, there are relations between objects. The following relations are defined in the ROM: constraint, connection, and predicate.

Constraint (ξ) is a descriptive, limiting, or particularizing relation of one object to another. It is represented by an arrow with a dotted head, as is shown in Error: Reference source not found. The arrow always points to the object to be constrained. The object is optional, depending on different cases.

Figure 3 Constraint relation.

Connection relation (ι), represented by a dashed arrow as is shown in Error: Reference source not found, is to connect two objects that do not constrain each other.

Figure 4 Connection relation.

Constraint (ξ) is a descriptive, limiting, or particularizing relation of one object to another. It is represented by an arrow with a dotted head, as is shown in Error: Reference source not found. The arrow always points to the object to be constrained. The object is optional, depending on different cases.

Figure 5 Constraint relation.

Type		Graphic Representation	Description
Object	Object		Everything in the universe is an object.
	Compound Object		It is an object that includes at least two objects in it.
Relations	Constraint Relation		It is a descriptive, limiting, or particularizing relation of one object to another.
	Connection Relation		It is to connect two objects that do not constrain each other.
	Predicate Relation		It describes an act of an object on another or that describes the states of an object.

Table 1: Objects and Relations

an expensive tool	
the riveting tool	
a tool in the box	
the cost of the tool	
location at the start	

Table 2: Examples of Constraint Relation

establish, implement, and maintain	

from one location to another	
linings onto shoes	

Table 3: Examples of Connection Relation

Transformation Rules

linings

As Chomsky explains in Syntactic Structure, a generative grammar is, a rule system formalized with mathematical precision that generates, without drawing upon any information that is not represented explicitly in the system, the grammatical sentences of the language that it describes and assigns to each sentence a structural description, or grammatical analysis.

A phrase structure(PS) grammar starts with a sentences and defines its parts, then define each of the sub-parts, continuing the redefinitions until a sentence has been produced, a PS grammar of English might be as follows:

S -> NP + VP
NP -> Mod + N (PP)
Mod -> (Art) (Adj)
VP -> V (ADV)
VP -> Aux + V
V -> Vi
V -> Vt + NP
V -> Vc + Adj
V -> Vc + NP
ADV -> PP
ADV -> Adv

PP -> Prep + NP
N -> bird, tree, boy,…
Art -> a, the,…
Adj -> dark, good, tall,…
Adv -> fast, slowly,…
Prep -> near, from, after,…
Vi -> cry, swim,…
Vt -> hit, break, eat,…
Vc -> is, become,…
Aux -> has, must, can,…

Where the abbreviations are:

S	sentence	Adv	adverb
NP	noun phrase	V	verb
VP	verb phrase	Aux	auxiliary
Mod	modifier	Vi	verb (intransitive)
Det	determiner	Vt	verb (transitive)
Art	article	Vc	verb (copulative)
Adj	adjective	PP	prepositional phrase
ADV	adverbial phrase	Prep	preposition

For example:

The dog can run in the park

This sentence was produced by applying the following rules:

S -> NP + VP

NP -> Det + N (PP)

N -> dog

VP -> VP + ADV

VP -> Aux + V

Aux -> can

V -> Vi

Vi -> run

ADV -> PP

PP -> Prep + NP

Prep -> in

NP -> Det + N

Det -> the

N -> park

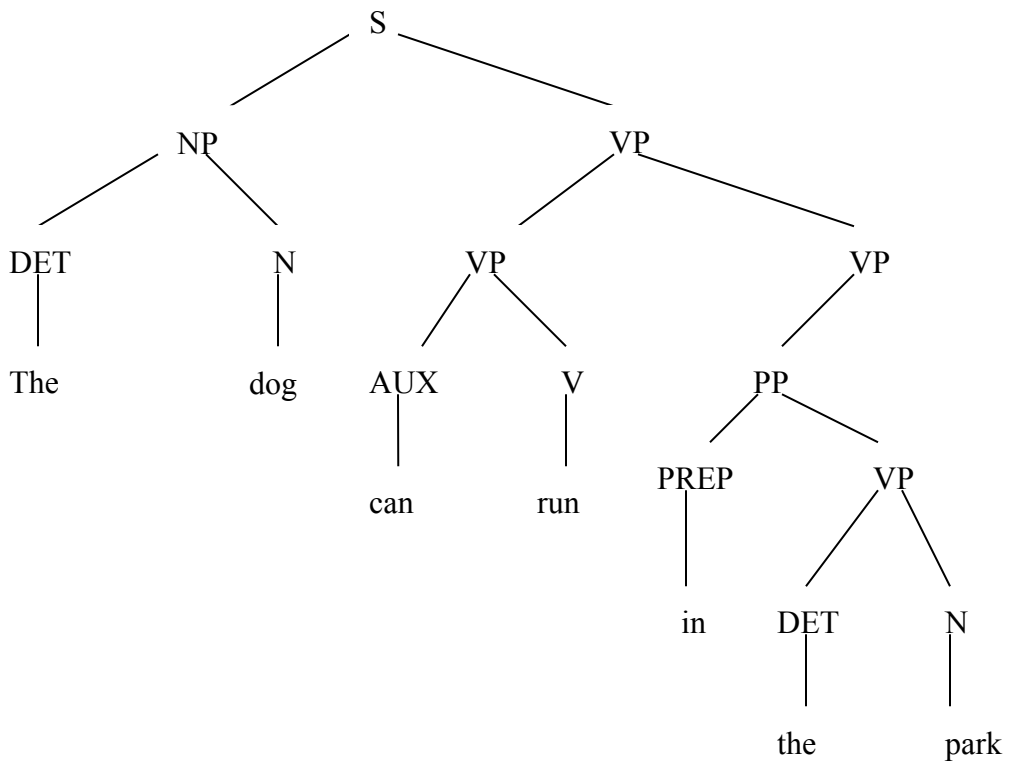
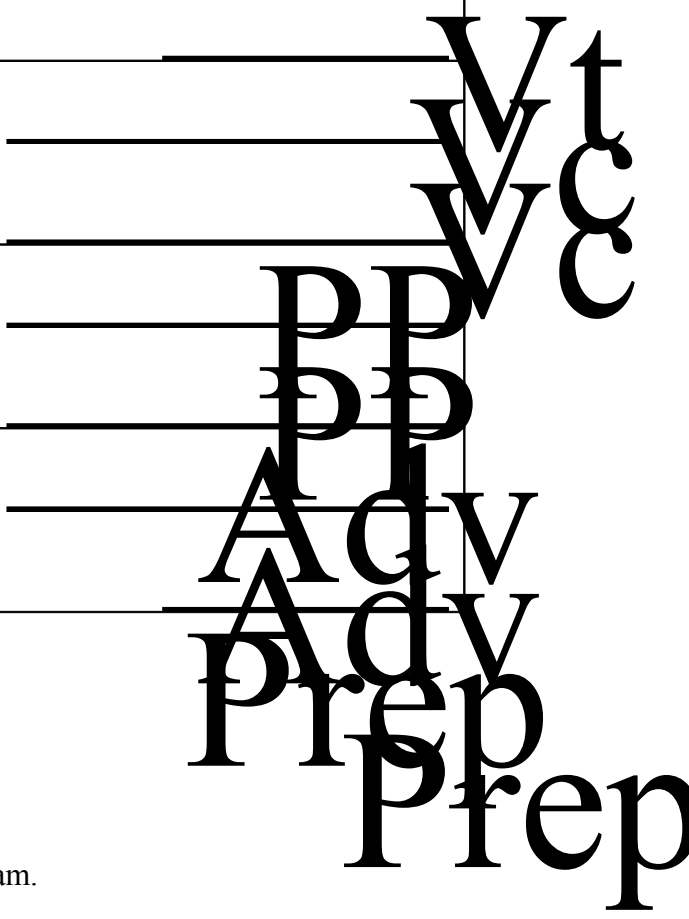


Figure Phrase Marker

Transform the phrase structure grammar to ROM symbol

Phrase Structure Grammar	**ROM Symbol**
S -> NP + VP	
NP -> Mod + N (PP)	
Mod -> (Art) (Adj)	
VP -> V (ADV)	
VP -> Aux + V	
V -> Vi	

V -> Vt + NP	
V -> Vc + Adj	
V -> Vc + NP	
ADV -> PP	
ADV -> Adv	
PP -> Prep + NP	



4.3 Scope

The scope of the project will include:

1. Identify the functionalities of our program.

2. Create ROM test plan based on the requirements
3. develop the class diagram
4. create sequence diagram for the following functionalities

   a) Convert a natural language file to ROM diagram,

   b) Display ROM diagram,

   c) Edit ROM diagram,

   d) Merge two ROM diagrams.

5. Develop ROM application based on the previous specification.

4.4 Team Planning

The team will complete the following activities to attaint the project goal.

- Identify project requirements
- Identify customer requirements
- Generate ideas
- Create use cases
- Review use cases
- Create test plan
- Develop class diagram
- Develop sequence diagram
- Review diagrams
- Develop the code
- Test code
- Resolve bugs
- Regression test

4.5 Execution Planning

4.5.1 Work Break down structure

A Gantt chart is used to monitor the work break down structure.

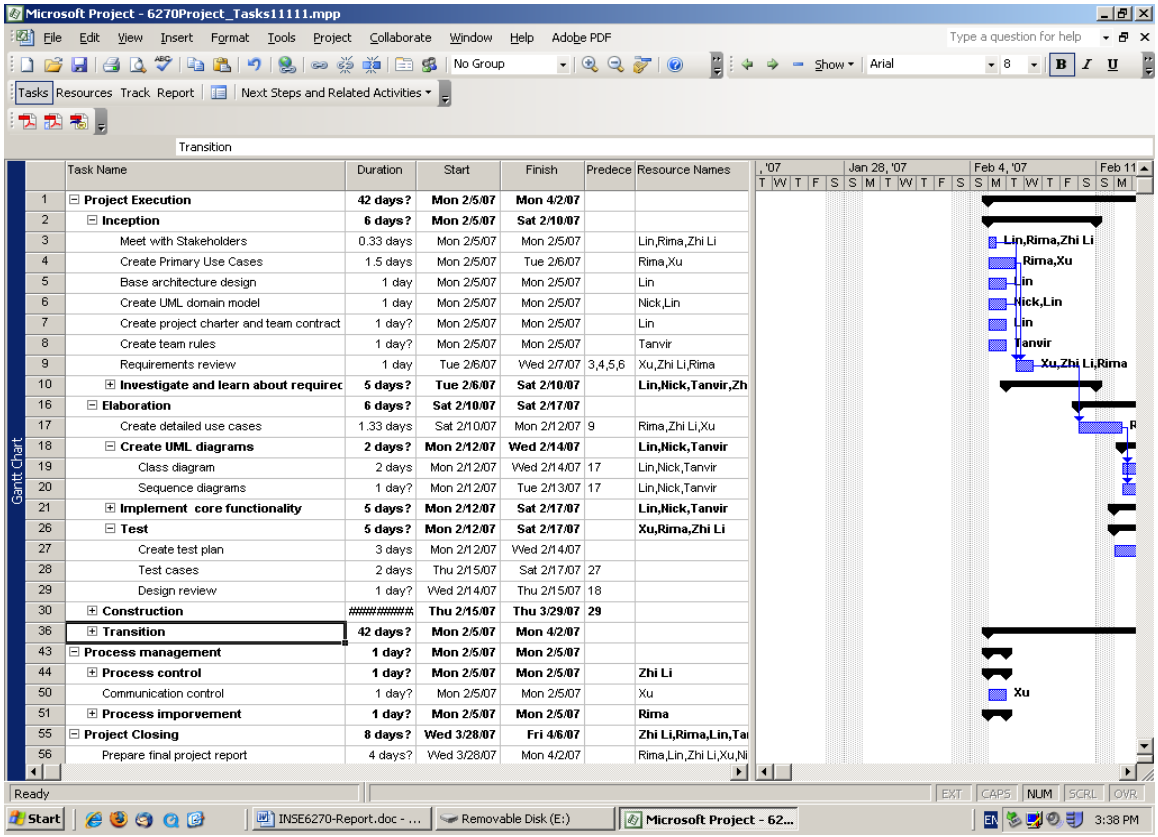


Figure: Gantt Chart

4.5.2 Resources

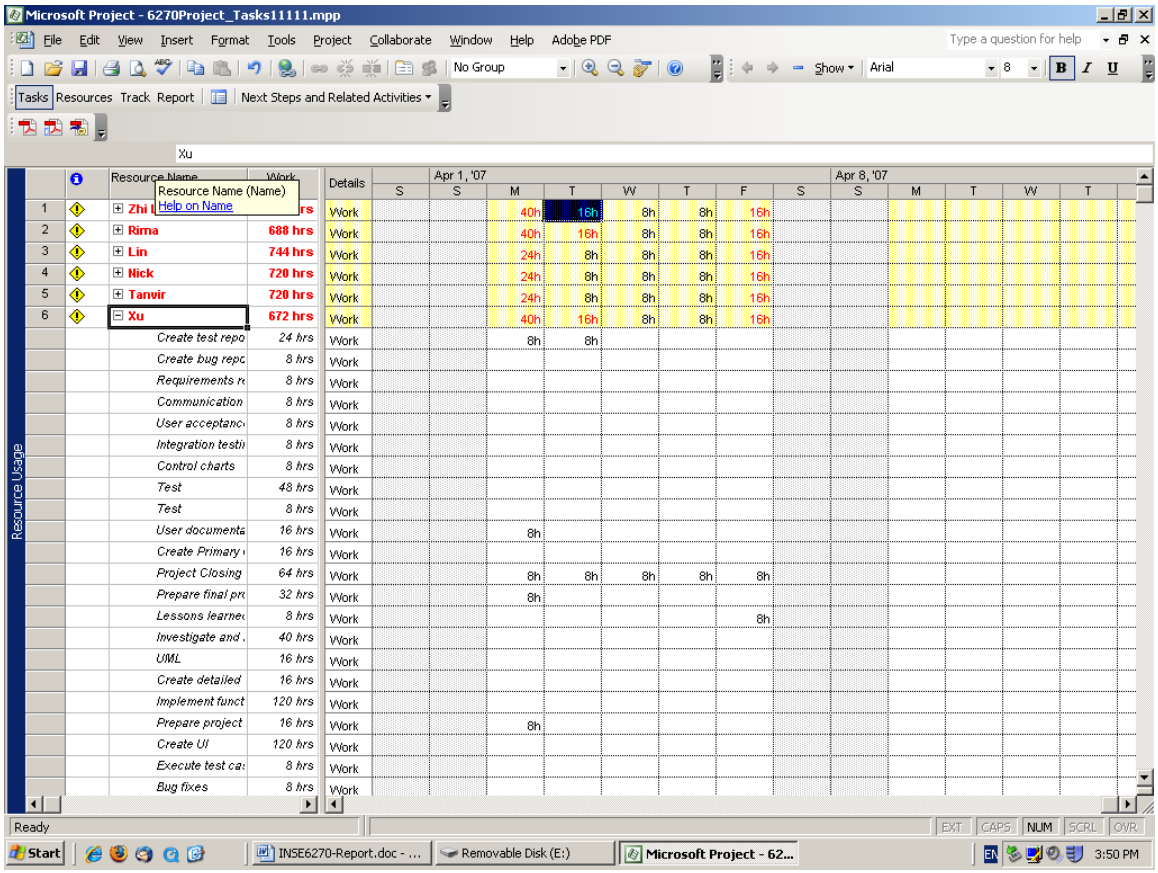
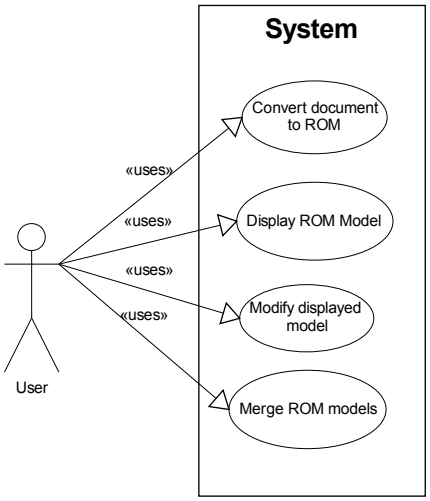


Figure: Resource Chart

4.6 Product Development

4.6.1 Use case diagram



4.6.2 Use cases

4.6.2.1 Use case: Convert Document

Name: Convert Document

Scope: The system under design
Level: User goal
Primary Actor: User
Stakeholders:

- Customer – Wants efficient and accurate document conversion
- Team members – Wants to ensure tool is functioning properly

Preconditions:

- Document written in natural language

Postconditions:

- Document is converted and represented as ROM model

Main Success Scenario:

1. User starts ROM application
2. User begins document conversion process
3. System asks user for document to convert
4. User locates and identifies document
5. System validates document format
6. System converts document into ROM representation
7. System displays model
8. User saves model
9. User closes application

4.6.2.2 Use case: display ROM diagram

Name: Display Model

Scope: The system under design
Level: User goal
Primary Actor: User
Stakeholders:
- Customer – Wants to be able to see model visually
- Team members – Wants to ensure tool is functioning properly

Preconditions:
- Existing ROM model representation

Postconditions:
- ROM model representation is visually displayed in a graphical format

Main Success Scenario:
1. User starts ROM application
2. User chooses to open existing ROM model
3. System prompts user to select model to display
4. User locates and identifies model to display
5. System analyzes model representation
6. System displays model

Extensions:
1a. System receives model representation from Convert Document process.

4.6.2.3 Use case: edit ROM diagram

Name: Edit Model

Scope: The system under design
Level: User goal
Primary Actor: User
Stakeholders:
- Customer – Wants to be able to edit new and existing models
- Team members – Wants to ensure tool is functioning properly

Preconditions:
- New or existing model is displayed

Postconditions:
- Modified model is correctly displayed and its data representation is correct

Main Success Scenario:
1. User selects object in visual model

2. System highlights selection
3. User modifies properties
4. System updates visual display
5. System updates data representation of model

Extensions:

1a. User creates new object in model
   1. User defines properties and relations of object
   2. System updates visual display
   3. System updates data representation of model

3a. User chooses to delete object
   1. System removes object and all associated relations
   2. System updates visual display
   3. System updates data representation of model

4.6.2.4 Use case: merge two ROM diagram

Name: Merge Models

Scope: The system under design
Level: User goal
Primary Actor: User
Stakeholders:

- Customer – Wants to be able to merge two existing models into one
- Team members – Wants to ensure tool is functioning properly

Preconditions:

- New or existing model is displayed
- Another model exists that is not currently displayed

Postconditions:

- Models are merged into one product model

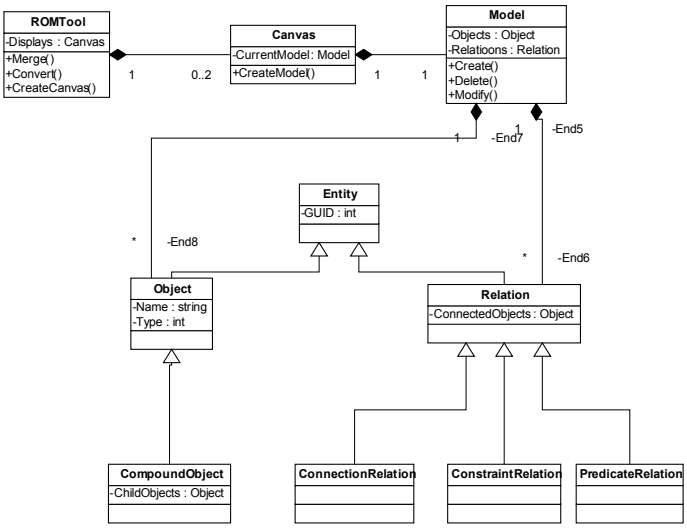
Main Success Scenario:

1. User selects to merge model with another
2. System prompts user to select first model
3. User locates and identifies first model
4. System prompts user to select second model
5. User locates and identifies second model
6. System validates model format
7. System asks user merge method
8. User selects automatic mode
9. System analyzes both models
10. System combines models into one representation
11. System displays product model

Extensions:

6a. User selects manual mode
1. System displays second model beside first model

4.6.3 Class diagram



4.6.4 Sequence diagram

4.6.5 Code

5 Control Process

5.1.1 Test plan

5.1.1.1 What will be tested

Testing will consist of several phase, each phase may or may not include testing of anyone or more of the following aspects of the ROM system (listed alphabetically):

- Accessibility
- Audit
- Availability
- Coding standards
- Compatibility
- Content
- Functional
- Legal
- Marketing
- Navigation
- Performance
- Reliability
- Scalability
- Security
- Site recognition
- Usability

The following is a list of functions that will be tested:

- Open/close file
- Select file
- Creation file
- Save file
- Convert document to ROM
- Display ROM Model
- Modify displayed model
- Merge ROM models
- Security features
- Error messages
- Start system
- Quit system
- ROM models Printing

5.1.1.2 What will not be tested

It is the intent that all of the individual test cases contained in each test plan will be performed. However, if time does not permit, some of the low priority test cases may be dropped.

5.1.1.3 Approach

The philosophy of the testing is risk-based testing, i.e. each test case will be prioritized as, High, Medium, or Low priority and then scheduled accordingly (Highest first). Exceptions to this general rule might include instances where:

- A large number of low priority test cases can be executed using a small amount of resources

- Scheduling conflicts arise
- A lower priority test is a pre-requisite of another higher priority test e.g. an expensive and high priority usability test might necessitate many of the inexpensive low priority navigational tests to have passed

Due to the lack of comprehensive requirements, navigational and functional tests may be scheduled first, so as to allow the testers the opportunity to gain familiarity with the Web site (thereby also allowing them to developing pseudo requirements).

The testing will use a combination of manual and automated testing, due to the limited duration of the testing; only automated tools that are already familiar to the system or have a minimum learning curve will be used.

Due to the short period of time allotted for test execution, the Web site's source code will be frozen while being tested. Except for critical fixes that are blocking the testing efforts, changes will not be scheduled while a unit of code is being tested.

Basic metrics will be kept for test effort (i.e. hours), test cases executed, and incidents. Due to the lack of available tools and time, no attempt will be made to collect more sophisticated metrics such as code coverage.

5.1.1.4 Test deliverables

a) Test plan

b) Test design specifications

c) Test cases

d) Test results

e) Bug reports

5.1.1.5 Risks and contingencies

Any risks associated with the testing of this project, include schedule, technical, management, personnel and requirements.

a) Schedule
Since the time is very short, the schedule for each phase is very aggressive and could affect testing. A slip in the schedule in one of the other phases could result in a subsequent slip in the test phase. Close project management is crucial to meeting the forecasted completion date.

b) Technical

Since this is a new Artificial Intelligence (AI) system, some concepts are not very mature. We will run our test on the web site, and depend on the theory of transferring the Natural Language to ROM. So the biggest technical risk is from the theory maturity, and the second risk is the web site.

c) Management
Management support is required so when the project falls behind, the test schedule does not get squeezed to make up for the delay. Management can reduce the risk of delays by supporting the test team throughout the testing phase and assigning people to this project with the required skills set.

d) Personnel
Due to the aggressive schedule, it is very important to have experienced testers on this project. Unexpected turnovers can impact the schedule. If attrition does happen, all efforts must be made to replace the experienced individual

e) Requirements
The test plan and test schedule are based on the current Requirements Document. Any changes to the requirements could affect the test schedule and will need to be approved by the team.

In the project the quality risk could be categorized as following:

Quality Risk Category	What Kind of Problems Fit Into this Category
Functionality	Failures that cause specific features not to work.
Load, Capacity, and Volume	Failures in scaling of system to expected peak concurrent usage levels.
Reliability/Stability	Failures to meet reasonable expectations of availability and mean-time-between-failure.
Stress, Error Handling, and Recovery	Failures due to beyond-peak or illegal conditions (i.e., knock-on effects of deliberately inflicted errors).
Date Handling	Failures in date math and handling.
Competitive Inferiority	Failures to match competing systems in quality.
Operations and Maintenance	Failures that endanger continuing operation, including backup/restore processes.
Usability	Failures in human factors, especially at the user interface.
Data Quality	Failures in processing, storing, or retrieving data.
Performance	Failures to perform as required under expected loads.
Localization	Failures in specific localities, including language, dictionary/thesaurus, and messages.
Compatibility	Failures with certain supported browser/OS combinations.
Security/Privacy	Failures to protect the system and secured data from fraudulent or malicious misuse.
Installation/Migration	Failures that prevent or impede deploying the system.

Documentation	Failures in operating instructions for users or system administrators.
Interfaces	Failures in interfaces between components.

5.1.1.6 Test Case and Bug Tracking

5.1.1.6.1 Test case tracking

Test cases will be tracked using a hierarchical collection of Excel worksheets. These will provide both detail level and summary level information on test cases.

Column Heading	Meaning
Priority	The priority of the test case.
State	The state of the test case. The possible states are: **Pass:** The test case concluded successfully. **Warn:** The test case concluded with an error, which the "Some IA Maker" management team has either deferred, closed as external to the product, or closed as unavoidable. **Fail:** The test case revealed a requirements or non-requirements failure that development will address. **Closed:** The test case previously revealed a failure that is now resolved. **In Queue:** The test remains to be executed (indicated by a blank in the column). **Skip:** The test will be skipped (explanation required in "Comment" column). **Blocked:** The test can not be run (explanation required in "Comment" column).
System Configuration(s)	A cross-reference to a worksheet that tracks specific configurations of client and server systems.
Bug ID	If the test failed, the identifier(s) assigned to the bug by the originator in "Some Bug Tracker".
Bug RPN	The risk priority number (severity times priority) of the bug(s), if applicable.
Plan Date	The planned date for the first execution of this test case.
Actual Date	The actual date it was first run.
Comment	Any comments related to the test case, required for those test cases in a "Skip" or "Blocked" state.

As the test organization runs each test, the state of each case will change from "In Queue" to one of the other states noted in the above table.
For each test that identifies a problem and enters a "Fail" or "Warn" state, the tester will open a bug report in "Some Bug Tracker".

5.1.1.6.2 Bug tracking

Test cases will be tracked using a hierarchical collection of Excel worksheets. These will provide both detail level and summary level information on test cases.

Field	**Meaning**
Bug ID	A unique identifier for each bug.
Severity	Technical problem severity, as follows: **1**. Critical failure; **2**. Non-critical failure; **3**. Cosmetic; **4**. Suggestion.
Priority	The end-user priority of the issue or the item that failed: **Must-fix**. Highest priority, must-fix for release; **Candidate**. Medium priority, desirable but not must-fix for release; **Enhancement**. Low priority, fix as time and resources allow; **Deferred.** Not for this release.
System Revision(s)	The revision names for each system affected by or possibly involved in causing the error.
Subsystem	The system most affected by the error, from the following list: [I need to update this.]
Date Opened	The date on which the bug report was opened.
State	The state of the issue, as follows: [TBD: Fix this to correspond to "Some Bug Tracker".] **Open:** The problem is deemed by the test engineer fully characterized and isolated; **Assigned:** The problem is accepted as fully characterized and isolated by development, and an owner, responsible for fixing the problem, is assigned; **Test:** Development have repaired the problem in some level of hardware or software, and someone owns testing the problem to evaluate the fix; **Closed:** The fix passed the test.
Originator	The name of the tester or other engineer who identified the problem, defaulting to the current user. For remotely generated reports, this will specify either the contact name or the outsource test organization itself.
Test ID	The test identifier (from the test-tracking matrix) corresponding to the test case the engineer ran that uncovered the issue. Also allowed are "Exploratory", "Other" and "Unknown".
Owner	The person responsible for moving the issue from "Assigned" to "Test" or from "Test" to "Reopened" or "Closed".

Field	Meaning
Abstract	A one- or two-sentence summary of the failure observed.
Remarks	A text field, free-format, consisting of two sections: **Steps to Reproduce:** A detailed, numbered process that will recreate the bug; **Isolation:** The steps performed to isolate the problem, including for reproducibility checking/statistics, bad-unit checking, and other pertinent tests.
Status	A free-from text field, consist of a "Comments" section and an "Actions" section, that updates what is happening to move the issue to closure (or deferral).
Closed Date	The date on which the issue was confirmed fixed or put on hold, which is used only when the issue is in a "closed" or "deferred" state.
Resolution	A free-format text field for a description of how the problem was resolved, which is filled in only when the issue is in a "closed" or "deferred" state.
Symptom	A classification of the symptom type from the standard "Some Bug Tracker" list.

When reporting a bug, test technicians and engineers shall adhere to the following process, taking detailed notes of all steps performed:

1. A bug report begins when a tester observes anomalous behavior on the part of the system under test, usually during the execution of the formal test case.
2. After reaching the logical conclusion of the failure, the tester repeats the steps required to reproduce the problem at least twice more, noting any variations in the behavior.
3. The tester then performs isolation steps as described below (see Error: Reference source not found).
4. The tester reports the bug, using "Some Bug Tracker", immediately after completing step three.
5. Prior to submitting the bug report, the tester finds a tester (preferably) or developer (alternatively) peer to review the bug report. Should the reviewer suggest any changes or additional tests, the tester will perform those steps and add the details to the report. Once the report is considered satisfactory by both originator and reviewer, the tester submits the report.

5.1.1.7 Resources and Responsibilities

The Test Leader and Project Manager will determine when system test will start and end. The Test leader will also be responsible for coordinating schedules, equipment, & tools for the testers as well as writing/updating the Test Plan, Weekly Test Status reports and Final Test Summary report. The testers will be responsible for writing the test cases and executing the tests. With the help of the Test Leader, test team will be responsible for the Beta and User Acceptance tests.

5.1.1.7.1 Resources

The test team will consist of:

- A Project Manager
- A Test Leader
- 2 Testers

5.1.1.7.2 Responsibilities

Project Manager	Responsible for Project schedules and the overall success of the project.
Lead Developer	Serve as a primary contact between the development team and the test team. Participate on CCB.
Test Lead	Ensures the overall success of the test cycles. Coordinate weekly meetings and communicate the testing status to the project team.
Testers	Responsible for performing the actual system testing.

5.1.2 Test cases

The Quality of our application is an important challenge that has a direct impact on our goal achievement. Our team is engaged to control the quality of our application, by performing test cases that investigate our code and verify that this code is error free.

5.1.2.1 Test case: Convert Document

Goal: Convert natural language into ROM Diagram

Preconditions: existent document written in natural language

Steps:

Step No.	Step Description	Expected Result	Actual result
1	User start ROM application	System ask user to choose option from 1 to 4	
2	User enter character ‘a’	System ask user to enter valid option	
3	User enter number not in the interval 1 to 4	System ask the user to enter valid option	
4	User enter 1 to convert the document	System ask user to enter the document path	
5	User enter invalid document path	System ask user to reenter valid document path	
6	User enter the document path containing invalid language	System ask user to reenter valid document	
7	User enter the document path containing the natural language	System display model and ask user if he wants to save the model	
8	User choose invalid response	System ask user to choose yes or no	
9	User choose Yes	System save the model	
10	User quit the application	System end	

Result: Document is converted and represented as ROM model

5.1.2.1 Test case: display ROM diagram

Goal: Display ROM Diagram

Preconditions: existent ROM Diagram

Steps:

Step No.	Step Description	Expected Result	Actual result
1	User start ROM application	System ask user to choose option from 1 to 4	

Step No.	Step Description	Expected Result	Actual result
2	User enter character ‘a’	System ask user to enter valid option	
3	User enter number not in the interval 1 to 4	System ask the user to enter valid option	
4	User enter 2 to display ROM Diagram	System ask user to enter ROM Diagram path	
5	User enter invalid document path	System ask user to reenter valid document path	
6	User enter path containing invalid document	System ask user to reenter valid ROM Diagram	
7	User enter the document path containing ROM Diagram	System display model	
8	User quit the application	System end	

Result: ROM model representation is visually displayed in a graphical format

5.1.2.1 Test case: Edit new ROM Model

Goal: Edit ROM Diagram

Preconditions: create New ROM Diagram

Steps:

Step No.	Step Description	Expected Result	Actual result
1	User start ROM application	System ask user to choose option from 1 to 4	
2	User enter character ‘a’	System ask user to enter valid option	
3	User enter number not in the interval 1 to 4	System ask the user to enter valid option	
4	User enter 3 to edit ROM Diagram	System ask user to enter option to edit an existing object or to create an new object	

Step No.	Step Description	Expected Result	Actual result
5	User enter 2 to edit new ROM Diagram	System ask user to enter properties and object relations	
6	User defines properties and relations of object	System updates data representation of model	
7	User select object	System highlights selection And ask user to modify or delete object	
8	User choose invalid response	System ask user to choose valid response.	
9	User choose to delete the object	System enable user to delete object selected	
10	User remove object and confirm the modification	System removes object and all associated relations and ask user if he want to save modification	
11	User choose invalid response	System ask user to choose yes or no	
12	User choose Yes	System save the model	
13	User quit the application	System end	

Result: ROM model is correctly modified and displayed

5.1.2.1 Test case: Edit existing ROM Model

Goal: **E**dit ROM Diagram

Preconditions: existent ROM Diagram.

Steps:

Step No.	Step Description	Expected Result	Actual result
1	User start ROM application	System ask user to choose option from 1 to 4	
2	User enter character ‘a’	System ask user to enter valid option	
3	User enter number not in the interval 1 to 4	System ask the user to enter valid option	
4	User enter 3 to edit ROM Diagram	System ask user to enter option to edit an existing object or to create an new object	
5	User enter 1 to edit existing ROM Diagram	System ask user to enter ROM Diagram path	
6	User enter invalid document path	System ask user to reenter valid document path	
7	User enter the document path containing invalid language	System ask user to reenter valid document	
8	User enter the document path containing ROM Diagram	System display model to modify	
9	User select object	System highlights selection And ask user to modify or delete object	
10	User choose invalid response	System ask user to choose valid response.	
11	User choose to modify the object	System enable user to modify the object selected	

Step No.	Step Description	Expected Result	Actual result
12	User modify the object and confirm the modification	System updates data representation of model and ask user if he want to save modification	
13	User choose invalid response	System ask user to choose yes or no	
14	User choose Yes	System save the model	
15	User quit the application	System end	

Result: ROM model is correctly modified and displayed

5.1.2.1 Test case: Merge manually ROM Model

Goal: ROM Diagrams merged into one model

Preconditions: New or existing model is displayed
Another model exists that is not currently displayed

Steps:

Step No.	Step Description	Expected Result	Actual result
1	User start ROM application	System ask user to choose option from 1 to 4	
2	User enter character ‘a’	System ask user to enter valid option	
3	User enter number not in the interval 1 to 4	System ask the user to enter valid option	
4	User enter 4 to merge ROM Diagram	System ask user to enter ROM ‘1’ Diagram path	
5	User enter invalid document path	System ask user to reenter valid document path	
6	User enter path containing invalid document	System ask user to reenter valid document	

Step No.	Step Description	Expected Result	Actual result
7	User enter valid document path containing ROM Diagram '1'	System ask user to enter ROM '2' Diagram path	
8	User enter invalid document path	System ask user to reenter valid document path	
9	User enter path containing invalid document	System ask user to reenter valid document	
10	User enter valid document path containing ROM Diagram '2'	System asks user to choose merge method	
11	User selects manual mode	System displays second model beside first model	
12	User quit the application	System end	

Result: *ROM* Models are merged into one product model

5.1.2.1 Test case: Merge automatically ROM Model

Goal: ROM Diagrams merged into one model

Preconditions: New or existing model is displayed
Another model exists that is not currently displayed

Steps:

Step No.	Step Description	Expected Result	Actual result
1	User start ROM application	System ask user to choose option from 1 to 4	
2	User enter character 'a'	System ask user to enter valid option	
3	User enter number not in the interval 1 to 4	System ask the user to enter valid option	
4	User enter 4 to merge ROM Diagram	System ask user to enter ROM '1' Diagram path	
5	User enter invalid document path	System ask user to reenter valid document path	

Step No.	Step Description	Expected Result	Actual result
6	User enter path containing invalid document	System ask user to reenter valid document	
7	User enter valid document path containing ROM Diagram '1'	System ask user to enter ROM '2' Diagram path	
8	User enter invalid document path	System ask user to reenter valid document path	
9	User enter path containing invalid document	System ask user to reenter valid document	
10	User enter valid document path containing ROM Diagram '2'	System asks user to choose merge method	
11	User selects automatic mode	System combines models into one representation And displays product model	
12	User quit the application	System end	

Result: *ROM* Models are merged into one product model

5.1.3 Self Control

The team members are in a state of self control, so they are responsible for the result of each task, in order to achieve the ultimate goal. Every team member must:

- Know what they are suppose to do
- know the schedule, cost and the specification of the project
- Analyze their performance
- Control their task and measuring its conformance and non-conformance to the specifications.

5.1.4 Optimizing self-control

Optimization of the self control can only be achieved by testing. The more test we perform, the better will be the self control and inspection of the application.

6 Improvement Process

6.1.1 Bug reports:

6.1.2 Corrective Action:

After brainstorming we found out the reasons of these failures, we performed correctives actions on the code in order to resolve those problems and then we execute recursive test on the application based on the same test case.

7 Conclusion

Appendix A – Meeting Minutes

Meeting Date:	Jan 13, 2007, Saturday
Meeting Time:	10:30AM-12:00AM
Meeting Members:	Jianqiang Lin, Yingbo Xu, Zhi Li, Tanvir Khan
Meeting Content:	Discuss about the project proposal
Come to Result:	Finish the project proposal

Meeting Date:	Jan 31, 2007, Wednesday
Meeting Time:	20:20PM-21:20PM
Meeting Members:	Jianqiang Lin, Nick Simmons, Rrima Ima, Tanvir Khan, Yingbo Xu, Zhi Li
Meeting Content:	Setup the scope for the project Discuss the development technique Brainstorming for the project management methodology Arrange the time for next meeting
Come to Result:	Develop the tasks outline and be prepared for WBS. Come out a decision using C# to program code, Google API & XML to do the interface between the database and application; using Internet dictionary to be the basic database.

Meeting Date:	Feb 3, 2007, Saturday
Meeting Time:	10:00AM-12:00AM
Meeting Members:	Jianqiang Lin, Nick Simmons, Tanvir Khan
Meeting Content:	Analyze the process activities Develop the tasks in detail Arrange the time for next meeting
Come to Result:	Develop the WBS draft

Meeting Date:	Feb 7, 2007, Wednesday
Meeting Time:	20:20PM-21:20PM
Meeting Members:	Jianqiang Lin, Nick Simmons, Tanvir Khan, Rrima Ima, Yingbo Xu, Zhi Li
Meeting Content:	Analyze the process activities Develop the WBS in detail (project charter, team contract, and quality control) Arrange the tasks for team members Arrange the time for next meeting
Come to Result:	Complete the Gantt Chart in detail.

Meeting Date: Feb 28, 2007, Wednesday
Meeting Time: 19:40PM-21:20PM
Meeting Members: Jianqiang Lin, Nick Simmons, Tanvir Khan, Yingbo Xu, Zhi Li,
Meeting Content: Combine the team control project report together
Come to Result: Finish the team control report.

Appendix B – Team Charter

Team Organization

Team Name	ROM software
Product Owner	Yong Zeng
Team Leader	Lin Jianqiang
Team Members	Nick Simmons, Tanvir Khan, Rima Alaoui, Yingbo Xu, Zhi Li

Team roles

Role	Members	Responsibilities
Project Manager	Jianqiang, Lin	Control the planning and management of the software project. Be a hierarchy of control starting from the project, reporting to the project board. Responsible for the time and budget control of the project.
Developer	Jianqiang, Lin Nick Simmons Tanvir Khan	Elaborate requirements into engineering specifications. Design system. Write code for the system efficiently and effectively.
Quality Assurance Analyst	Rima Alaoui Yingbo Xu Zhi Li	Perform quality process reviews. Analyze cause and effects of problems encountered with the team. Find and implement solutions for problems that lead to improvements.
Quality Control Technician	Rima Alaoui Yingbo Xu Zhi Li	Inspect intermediate and final work products. Verify conformance of products to requirements and suitability of processes - includes software testing and inspection. Measure quality attributes and provide feedback to other groups.

Standards

Process: Use the Rational Unified Process (RUP) model as a guideline.

Requirements: Use Cases for requirements specification.

Design: Use UML diagrams to illustrate system architecture and design.

Development: In C#, using XML.

Testing: Tests will be based off of Use Cases. Verify system conforms to requirements.