DREXEL ISCHOOL

Apartment Management System Analysis & Design

INFO 620 Information Systems Analysis and Design
Spring Quarter 2010

Nathan Vasserman
Fangwu Wei
David Fernandez
Andrew Messina

Final Report Submission
06/10/2010

INFO 620: Information Systems Analysis and Design, Spring Quarter 2010

Fangwu Wei, David Fernandez, Nathan Vasserman, Andrew Messina,

Project Category: Analysis & Design, Apartment Management System

Contents

Introduction

For our project, we have decided to do comprehensive UML model of the Apartment Management System. The requirements of the AMS require a tool be built for a local building management company wishing to automate many of the interactions between tenant, landlord and apartment management staff. In addition to just handling rent money exchange, the system needs to keep track of the entire services apartment owners offer to their tenants such as maintenance, basic inspection and transfer of tenants.

The project proved to be a large undertaking as we spent a significant amount of time delving into the details of what the maintenance an apartment building requires and all of the rent laws in Pennsylvania. The amount of work required significant breakdown by services. We had team members who worked on rent interactions, inspection processes, maintenance and the unfortunate possibility that a tenant's lease might be terminated, either by the tenants or the landlord's choice. The following design document reflects all of those features and more.

For the group members that have never lived in an apartment, this project proved to be quite the learning experience. We hope the following can accurately portray a sample of what such a software suite would require and how it could be coded to become a reality.

System Analysis

1. Title: Analysis and Design of an Apartment Management System

2. The Problem Statement

A small Apartment Rental company would like to create a management system, common for every apartment blocks distributed by Philadelphia and towns around.

(a) Overall goals of the system

The overall goals of the system are to keep track of tenant maintenance requests, tenant record, document and contract management, to make easier to the tenant and controlling the rental payment.

(b) Context and Importance of the system

It is critical that any apartment rental company to control the expenses of the apartment management and tracking the rental payment for the tenants. Managers complain that tenants often forget to pay the rent on time, and some of them are even difficult when it comes to communicating or being localized. An on-line system which improves the communication between property managers and tenants will serve as a reminder for making on-line payments obligations and in case of delays, and to warn them about it, instantly. Tenants complain that managers are slow in problem solving and sometimes they are difficult to localize. An on line system to make request about maintenance problems allows managers to be more effective to resolve the problem and the central management to be able to plan expenses, to contract or hire some services at the best price and put on disposition to very apartment manager the company which would help with the problem.

(c) Scope of the project

IN-Scope:

AMS will include only tenants and their requests and obligation, rental payments, apartment maintenance services as plumbers, windows, insects, etc., building maintenance service such as gardening, roof, central heating, etc, and contract management as new tenant contract, current tenant renewals. It also includes requests and reports from the managers to the central administration and service contract from the central administration to the managers.

OUT-Scope:

SAMS (Small Apartment Management System) will **not** include a central accounting system.

3. Requirements

3.1 Functional Requirements (partial list)

The system will be password-protected. AMS will be a multi-user system where every user must log in. AMS needs to perform the following functions:

Tenants to the manager system:

(1) Request a change of apartment.

(2) Request a maintenance petition.
(3) Complaints.
(4) Pay the rent on-line.

Manager System to the tenants

(1) Add a new tenant and make and managing his/her contract.
(2) Warn and report any tenant about his/her rental payment.
(3) Report any interesting information (new services, taxes, etc)
(3) Manage the tenant maintenance request, and reporting about it.

Manager System to a manager:

(1) Report about any tenant maintenance tasks.
(2) Report about any periodical building maintenance.
(3) Pick up the manager request to the central administration.

Manager System to Central Administration:

(1) Report about the tenant rent payments.
(2) Report about the maintenance services.
(3) Request available services.
(4) Report and send tenant contract or documents.

3.2 **Data requirements (Partial List)**

For clients, keep track of client's name, address, business phone, home phone, cell phone, outstanding balance, starting date, and business type. The business type is One of S-corp, C-Corp, Partnership, LLC, LLP, SolePropreiter, Estate, Trust, Non-Profit, Individual, Other.

For each billable item, SAMS will keep track of item#, date entered, description, initial amount, status, and balance. Billable items are either monthly service charge or other special service charge. For the latter case, the name and the fee of the service is recorded.

For each invoice, SAMS will need to keep track of invoice#, invoice date, total billing amount from all the billable items which are not marked as Paid In Full.

For each payment, SAMS will keep track of payment#, payment date, description, amount, payment type, check#, and bank name.

3.3 Business Rules and Logic **(Partial List)**

(1) The outstanding balance of a client will always reflect the summation of balances of all the billable items.
(2) When a new billable item is created, initial amount and balance are zero. Later when a payment is made, the initial amount remains the same, but balance must be reduced by the amount of payment amount.
(3) The status of billable items must be properly changed its value. Initially, when the item is created, the STATUS is Un-invoiced. When an invoice is sent out, the STATUS becomes invoiced. When the item is paid in full, the STATUS becomes Paid in-full. When the tenant is deceased or other circumstances arise, the STATUS will become payment-in-process.
(4) When the item is paid by only by partially, the STATUS becomes Paid-in-Partial. The state changes need to be automatic. A billable item could also be discounted or cancelled.
(5) The total billable amount is derived as the summation of current unpaid billable items.
(6) SAMS will be used by multiple accountants, and thus some important activities must be noted on who recorded or changed the record with the last update date.
(7) When a request for a sublease is sent out, the system will then process the request. With regard to the information the system will either approve or decline the request.

3.4 Non-functional requirements

Requirements on usability, reliability, performance, supportability, security, recovery, interface, implementation, operation, and legal.

(1) The system will be a screen-based application.
(2) Menus should be organized in a hierarchical manner (usability)
(3) The system will be password-protected. (Security)
(4) SAMS will be backed up daily. (Back up)

3.5 Other Assumptions

(What are the assumptions of the system? What are HW and SW constraints? Are there any implementation constraints?)

(1) We will assume SAMS will be used by a small accounting firm in a real-world setting.
(2) SAMS runs on a client/server environment, running Windows Server as OS.

(3) The underlying DB system is Microsoft Access
(4) State specific rental laws will be based on Pennsylvania laws, should any discrepancies arrive.
(5) Buildings affected will not be rent-controlled units.

4. Examples of system input/output, etc.

Examples of system input:

(1) A tenant pays rent by a personal check.
(2) A tenant wants to sublease an apartment before completing the contract.
(3) A tenant rents an apartment.
(4) A staff adds new tenant information.

Examples of system output:

(1) System prints tenant information including payment history by tenant requests.
(2) Payment reports comply with local tax codes.
(3) System keeps logs of rental unit history. Past and present tenants.
(4) System maintains information regarding regular unit inspections and compliance with tenant; insuring units remain up to code.

5. Knowledge Acquisition

The problem is an Analysis and Design project. First we will develop our requirements based on our common sense and the current knowledge. After that, we will consult with an actual accounting office to validate our requirements.

6. Software and/or hardware involved

We will use Rational Rose for developing all the UML diagrams. Microsoft Access will be used when the system needs to be prototyped to get the ideas for screen developments. The application itself is PC-based running on XP.

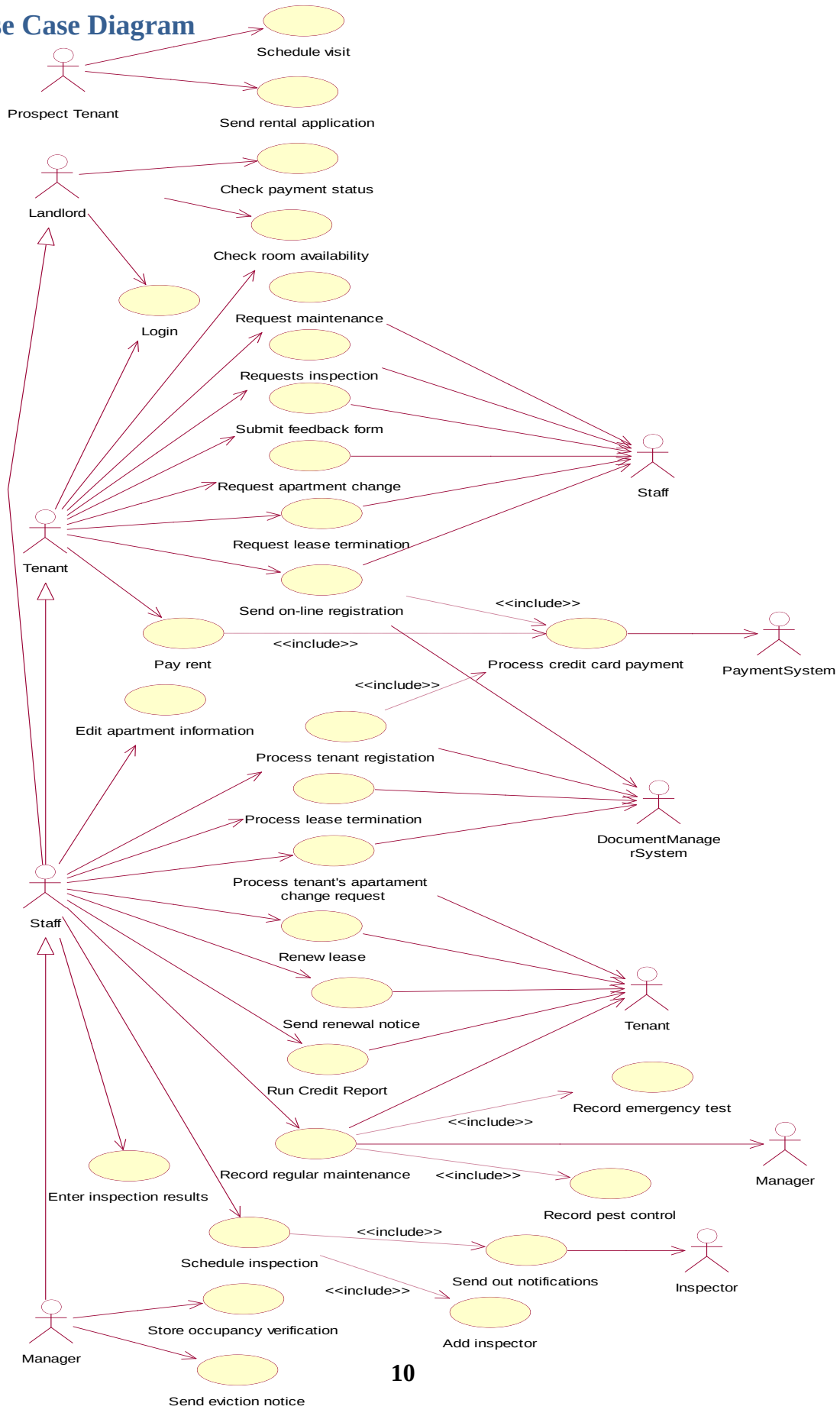
7. Proposed Deliverables and work plans

We intend to turn in a complete set of UML diagrams along with supporting documentation. We will also put together a report describing our experience with analyzing the current process, what we were able to learn from our study, known pitfalls, remaining questions after project, and any recommendations on how to improve the current system.

8. Known References (so far)

References at this point will be drawn from personal experiences and widely available resources on laws and regulations regarding residential rental units. Most team members have lived in an apartment building for at least a portion of their lives. Pennsylvania state law has rules regarding the treatment and maintenance of a rental unit, and the rights and responsibilities of a tenant. Codes can be found here for http://www.rentlaw.com/pennsylvaniarentlaw.htm Pennsylvania laws. Laws differ from state to state but for this project we will assume PA regulations.

Use Case Diagram



Use Case Descriptions

Use Case Name:	Schedule visit
Actor(s):	Prospect Tenant
Purpose:	Booking landlord time for visiting the apartments.
Overview:	Prospect tenant (after a visitor phone petition) chooses a date and time available for visiting the apartments.

Use Case Name:	Send rental application
Actor(s):	Prospect Tenant
Purpose:	Renting an apartment
Overview:	Prospect tenant can rent an apartment sending the solicitation form and required digital documents. Prospect tenant must provide a credit card to pay the security deposit and prepaid rent.

Use Case Name:	Check payment status
Actor(s):	Landlord, Staff, Manager
Purpose:	To clearly know the payment status of an apartment.
Overview:	Landlord can check the payment status to know whether the tenant pays the rent or the apartment payment is on time.

Use Case Name:	Check room availability
Actor(s):	Landlord, Tenant, Staff, Manager
Purpose:	To check the apartment availability and basic information.
Overview:	Landlord can check whether the apartment is available and view the basic information related to the apartment.

Use Case Name:	Request inspection
Actor(s):	Tenant, Staff, Manager
Purpose:	To submit a request to inspect the building.
Overview:	Shows the process of requesting inspections. Tenant will submit the request in order to be processed by the landlord.

Use Case Name:	Request maintenance
Actor(s):	Tenant, Staff, Manager
Purpose:	To submit a request to fix accidental apartment problems.
Overview:	Shows the process of requesting maintenance. Tenant will make an appointment, set a schedule, and fill out a maintenance form for repairing the accidental maintenance problems.

Use Case Name:	Submit feedback form
Actor(s):	Tenant, Staff, Manager
Purpose:	To provide a real-time feedback.
Overview:	After the accidental maintenance, tenants will fill out a feedback form and submit it. This form will help apartment managers improve their work.

Use Case Name:	Request an apartment change
Actor(s):	Tenant, Staff, Manager
Purpose:	Requesting the landlord for moving from the present apartment to another
Overview:	Tenant chooses a new apartment and the date to move and send the solicitation to the landlord for studying

Use Case Name:	Request lease termination
Actor(s):	Tenant, Staff, Manager
Purpose:	Requesting the landlord for moving out from the present apartment and finishing the lease.
Overview:	Tenant report landlord the date to move out.

Use Case Name:	Pay rent
Actor(s):	Tenant, Staff, Manager
Purpose:	Allows Customer to make payments online.
Overview:	Customers use the AMS to pay the rent.

Use Case Name:	Login
Actor(s):	Tenant, Landlord, Staff, Manager
Purpose:	To use different levels of security access to protect user's information.
Overview:	Based on the different security levels of users, the system only provides proper information to users.

Use Case Name:	Edit apartment information
Actor(s):	Staff, Manager
Purpose:	To manage apartment information.
Overview:	Staff or manager check/update apartment information, such as rental fee.

Use Case Name:	Process tenant registration
Actor(s):	Staff, Manager
Purpose:	Renting an apartment for a new tenant
Overview:	Landlord enters the entire tenant's data and the Document Manager System is sent all the necessary data to generate the lease.

Use Case Name:	Process lease termination
Actor(s):	Staff, Manager
Purpose:	Releasing the apartment, calculating the amount the former tenant will get or pay and making the Document Manager System know about the lease termination.
Overview:	Landlord enters any damage to the apartment and the apartment conditions or required services to be in perfect conditions. The system calculates the former tenant's final balance.

Use Case Name:	Renew the lease
Actor(s):	Automatic process
Purpose:	Report to the tenant the lease renewal and any increase in the rent.
Overview:	70 days before the lease expires, the system report to the tenant, the lease will be automatically renewed and the new rent.

Use Case Name:	Send renewal notice
Actor(s):	Staff, Manager
Purpose:	To send an email to notify tenant that the lease is expiring.
Overview:	Staff or manager sends an email to remind tenant to renew the apartment lease.

Use Case Name:	Run Credit Report
Actor(s):	None, everyday process
Purpose:	Keep tenants reported about the payment status.
Overview:	Runs a credit report on tenants to ensure that all tenants have settled their debts and are able to pay rent, report about fines for lateness, etc.

Use Case Name:	Process tenant's apartment change request
Actor(s):	Staff, Manager
Purpose:	Accepting a tenant's apartment change request
Overview:	The landlord accepts the change petition, so a new lease must be signed.

Use Case Name:	Record regular maintenance
Actor(s):	Staff, Manager
Purpose:	To make sure each tenant knows the maintenance schedule.
Overview:	An email about regular seasonal/annual maintenance will be sent to all the tenants in order to notify tenants in advance for the inconvenience, so they can make a slight change for their schedule.

Use Case Name:	Schedule inspection
Actor(s):	Staff, Manager
Purpose:	To program an external inspection of the building.
Overview:	Landlord selects an external inspector and fixes the inspection date and time. The inspection is notified to the tenants.

Use Case Name:	Enter inspection results
Actor(s):	Staff, Manager
Purpose:	To enter the inspection result in order to make them know to the tenants.
Overview:	Landlord enters the inspection results to the system. Tenants can also pull out the inspection results from the system.

Use Case Name:	Store occupancy verification
Actor(s):	Manager
Purpose:	To verify rental applications entered by staff.
Overview:	Allows the manager to verify rental occupation, cost and profits.

Use Case Name:	Send eviction notice
Actor(s):	Manager
Purpose:	To send an email about eviction notice.
Overview:	Manager checks the apartment and tenant status, and then sends an eviction notice.

Use Case Name:	Record pest control
Actor(s):	
Purpose:	To eliminate pest to make apartments cleaner.
Overview:	Landlord will regularly (seasonal) eliminate pest, including mice, cockroaches, and bugs.

This is an included use case. So, it does not have a responsible actor.

Use Case Name:	Record emergency test
Actor(s):	
Purpose:	To test the facilities to make apartments safer.
Overview:	Landlord will regularly test the fire alarm/sprinkler and make the facilities usable.

This is an included use case. So, it does not have a responsible actor.

Apartment Unit Assumptions

Pennsylvania Landlord/Tennant laws can be found here:
http://www.rentlaw.com/pennsylvaniarentlaw.htm

Actors: Tenant, Landlord, Inspector, Superintendent

Assumptions:

1) Apartment inspections are done annually.
2) Inspections are performed by a 3rd party.

3) The 3rd party, known as the "inspector" report's findings in a report to the superintendent and landlord.
4) Apartments must be found up to code and in good condition and livable. If an inspection finds them to not be so, the landlord must discuss with the tenant what can be done to improve them, and who is liable for repairs.
5) Tenants are notified at least 2 weeks in advance before an inspection is scheduled.
6) A tenant must be present during the inspection.
7) Inspection fees are paid by the landlord.
8) The landlord will share the inspection results with the tenant after the inspection is complete.
9) The superintendent is allowed to be present during the inspection to verify the results.
10) A tenant may request an additional inspection up to once a year, if they feel something is wrong with the property.
11) The system will send the inspection results to the landlord, the landlord may send it to the tenant.
12) The inspector is chosen by the system, which finds the most available inspector at the time of scheduling.
13) Inspections are scheduled within the system by the landlord, notifications are sent out to all parties.
14) Inspection results are entered into the computer system by the superintendent.
15) If an apartment is not passed for inspection, the system will flag it, and it will be investigated by the landlord.
16) Regular maintenance will be performed by the superintendent on an as needed basis or by the request of the tenant with approval of the landlord.
17) Sprinkler tests and pest control will require notification to the tenant at least 2 weeks in advance.
18) Tenants are provided with AMS system feedback forms for their maintenance jobs at their request. They are then submitted to the system for review.
19) Tenants will be provided with a single login for maintenance and payment capabilities through the system.

Rules for Payment, Leases and new tenants

Actors: Tenant, Landlord, Payment System, Document Management System, Manager

Assumptions:

1) All payments are processed online through the payment system
2) Payment System accepts all major credit cards and debit cards.
3) Payment system is informed of credit reports, to determine if a tenant is a potential credit risk.

4) The managers may report to the system with information about credit reports, risks and rental occupation.
5) Rent is due on the 1st of the month.
6) Before leases can be accepted or renewed, the landlord must manually view a credit report.
7) Probable causes for lease termination, induced by the landlord, include: non-payment of rent after 60 days, tenant requests termination, extensive destruction to the property by the tenant, illegal activity going on inside an apartment, housing of pets strictly forbidden by the lease. Other extenuating circumstances are handled on a case-by-case basis with involvement of the tenant, landlord and possibly the courts.
8) In the event of a non-consensual lease termination, the legal burden will be on the landlord to begin the eviction process through the sheriff's office. The landlord will flag the lease as an eviction in the AMS.
9) Old Tenant information is kept in the system archived for a period of 7 years, then purged from the database.
10) Apartment changes must be submitted into AMS with an extensive written description of why a tenant or landlord is requesting the change. Changes must be approved by both parties in the AMS.

Detailed Use Case Descriptions

USE CASE #	1	
USE CASE Name	Record regular maintenance	
ACTOR	Staff, Manager	
Goal (1 phrase)	To make sure each tenant knows the maintenance schedule	
Overview and scope	An email about regular seasonal/annual maintenance will be sent to all the tenants in order to notify tenants in advance for the inconvenience, so they can make a slight change for their schedule. Meanwhile, the related records will be stored in the system.	
Level	Primary	
Preconditions	The staff is logged into the system. The tenants' email list is available.	
Postconditions	The information (name and date) of regular maintenance was recorded and stored. The notification email with the information of regular maintenance was sent. The notification email is forwarded to manager.	
Trigger	The staff has chosen "Regular Maintenance" option.	
Included Use Cases	Record pest control. Record emergency test.	
Extending Use Cases	None	
MAIN SUCCESSFUL SCENARIO in numbered sequence	**Actor Action**	**System Action**
	1. The staff selects the "Record a new regular maintenance".	2. AMS displays a form to fill out the emergency test information.
	3. The staff enters the date and type of testing device. And then, the staff submits all the data.	4. AMS confirms the data entered.
		5. **INCLUDE** Record emergency test.
		6. AMS displays a form to fill out the pest control information.
	7. The staff enters date, type of pest, and type of pest control. And then, the staff submits all the data.	8. AMS confirms the data entered.
		9. **INCLUDE** Record pest control.
		10. AMS displays a complete notification of regular maintenance and also displays three buttons: "Confirm", "Back", and "Cancel".

	11. The staff confirms it.	12. The notification is stored to the database.
		13. AMS displays two options: "Send this now" and "Send this later".
	14. The staff selects "Send this now".	15. AMS displays the list of email addresses.
	16. The staff checks "Select all current tenants", and then submits.	17. AMS sends the notification to current tenants and manager.
		18. AMS displays message "The notification is sent."
OTHER SUCCESSFUL SCENARIOS	**Step**	**Branching Action**
	6a. Required data is not entered.	AMS displays a message to enter the required data.
	10a. Required data is not entered.	AMS displays a message to enter the required data.
	13a. The staff chooses "Back".	Go to step 5.
UNSUCCESSFUL SCENARIOS	**Conditions**	**Actions**
	13b. The staff chooses "Cancel".	Abort the notification, display cancellation message, and go to the main screen.
	16a. The staff chooses "Send this later".	Return to the main screen.
	17a. Email bounces back because of invalid email address.	Ask to enter another address or abort
Priority in scheduling	Primary	
Frequency	2 per month	
Other non-functional requirements	1. The notification is sent in 10 seconds. 2. Emergency test information and pest control information are recorded in 5 seconds.	
Business rules and data logic	1. Emergency tests and pest control will require notification to the tenant at least 2 weeks in advance. 2. Regular maintenance is performed by the apartment engineers.	
Superordinates	None	
Developer	Fangwu Wei	
Creation date and last modified date	Version 1, 05/08/2010 Version 2, 06/02/2010	
Other Comments		

USE CASE #	2	
USE CASE Name	Record emergency test	
ACTOR		
Goal (1 phrase)	To test the facilities to make apartments safer	
Overview and scope	Apartment staff will regularly test the emergency devices (fire alarm and sprinkler) and make the facilities usable.	
Level	Included	
Preconditions	Emergency test information (date and type of testing device) is entered.	
Postconditions	Emergency test data was recorded. Notification related to emergency test was generated.	
Trigger	A new notification was written or an old notification was modified.	
Included Use Cases	None	
Extending Use Cases	None	
MAIN SUCCESSFUL SCENARIO in numbered sequence	**Actor Action**	**System Action**
		1. AMS fills out a notification template by using the data entered.
		2. AMS generates the notification of pest control.
		3. AMS stores the notification in the database temporarily.
OTHER SUCCESSFUL SCENARIOS	**Step**	**Branching Action**
UNSUCCESSFUL SCENARIOS	**Conditions**	**Actions**
Priority in scheduling	First	
Frequency	2 per month	
Other non-functional requirements	The notification is generated within 2 seconds.	
Business rules and data logic	The notification builder is embedded to the system.	
Superordinates	Record regular maintenance	
Developer	Fangwu Wei	
Creation date and last modified date	Version 1, 05/08/2010 Version 2, 06/02/2010	
Other Comments		

USE CASE #	3	
USE CASE Name	Record pest control	
ACTOR		
Goal (1 phrase)	To eliminate pest to make apartments cleaner	
Overview and scope	Apartment staff will regularly (seasonal) eliminate pest, including mice, cockroaches, and bugs by using different methods of pest control.	
Level	Included	
Preconditions	Pest control information (date, type of pest, and type of pest control) is entered.	
Postconditions	Pest control data was recorded. Notification related to pest control was generated.	
Trigger	A new notification was written or an old notification was modified.	
Included Use Cases	None	
Extending Use Cases	None	
MAIN SUCCESSFUL SCENARIO in numbered sequence	**Actor Action**	**System Action**
		1. AMS fills out a notification template by using the data entered.
		2. AMS generates the notification of pest control.
		3. AMS stores the notification in the database temporarily.
OTHER SUCCESSFUL SCENARIOS	**Step**	**Branching Action**
UNSUCCESSFUL SCENARIOS	**Conditions**	**Actions**
Priority in scheduling	First	
Frequency	2 per month	
Other non-functional requirements	The notification is generated within 2 seconds.	
Business rules and data logic	The notification builder is embedded to the system.	
Superordinates	Record regular maintenance	
Developer	Fangwu Wei	
Creation date and last modified date	Version 1, 05/08/2010 Version 2, 06/02/2010	
Other Comments		

USE CASE #	4	
USE CASE Name	Process Rental Payment	
ACTOR	Tenant	
Goal (1 phrase)	Allows Customer to make payments online.	
Overview and scope	A customer makes a rental payment through the web AMS system. The customer pays using a credit card. The system records relevant payment and transaction data, process the payment, and changes the payment status.	
Level	Primary	
Preconditions	The tenant logs into the AMS system. The customer then pays his balance with a credit card.	
Postconditions in words(write in passive and past tense)	The payment status was changed to reflect the Tenants payment. The payment information and transaction data was stored. A payment confirmation email was sent to the tenant.	
Trigger	A tenant makes a rental payment using a credit card	
Included Use Cases	Process Credit Card Payment	
Extending Use Cases	None	
***MAIN SUCCESSFUL SCENARIO* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	**Actor Action**	**System Action**
	1. A tenant makes a full rental payment via credit card.	2. The system processes the transaction and verifies the credit card.
		3. INCLUDE Process Credit Card Payment
		4. The System, records relevant transaction and payment data.
	5. The tenant's payment status is changed from INVOICED to PAID-IN-FULL.	6 A Confirmation Email is sent to the tenant

OTHER SUCCESSFUL SCENARIOS (Specify any ***successful*** variations of the *normal* execution path, including extension points using EXTEND *eus_name*)	**Step**	**Branching Action**
	1a. A tenant makes a partial payment via credit card.	The payment is recorded as PAID-IN-PARTIAL
		An Invoice is sent to the customer requesting the remaining amount
UNSUCCESSFUL SCENARIOS (*erroneous* situations*)*	**Conditions**	**Actions**
	3a. Credit Card Has Insufficient Funds	Abort the transaction
	6a. Email address is invalid	Send confirmation email to Staff
Priority in scheduling	Primary	
Frequency	Monthly	
Other non-functional requirements	1. The system will be a screen-based application. 2. Payment and transaction Information will be backed up daily 3. Menus should be organized in a hierarchical manner	
Business rules and data logic	1. Use of a Credit Card is required for online payment 2 The status of billable items must be properly changed following the transaction 3 The rental payment amount is generated before the customer makes payment.	
Superordinates	None	
Developer	Andrew Messina	
Creation date and last modified date	Created 4-30-10 Modified 5-2-10	
Other Comments		

USE CASE #	5	
USE CASE Name	Process Lease Termination	
ACTOR	Staff, Document Management System	
Goal (1 phrase)	To terminate a tenant's lease	
Overview and scope	A staff member will initiate the ending of a tenant's lease, either through cause of their eviction or decision to move out. Lease terminations, specifically evictions, must be justified for reasons such as non-payment of rent or breaking apartment rules.	
Level	Primary	
Preconditions	Tenant must have requested to end their lease or landlord must have documented proof for reason for eviction. Tenant must be otherwise be the primary lessee for the unit.	
Postconditions in words(write in passive and past tense)	Tenant was removed from apartment, the unit shows as vacant.	
Trigger	Staff requested by landlord or tenant to terminate a lease.	
Included Use Cases		
Extending Use Cases		
***MAIN SUCCESSFUL SCENARIO* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	**Actor Action**	**System Action**
	1. Staff, being asked by landlord to terminate lease, begins termination process.	2. Document management system retrieves current lease information
	3. Staff inputs reasons for termination of lease.	4. System confirms they are valid and lawful reasons for termination of a lease.
	5. Staff confirms	6. Lease is terminated, unit is labeled as vacant.

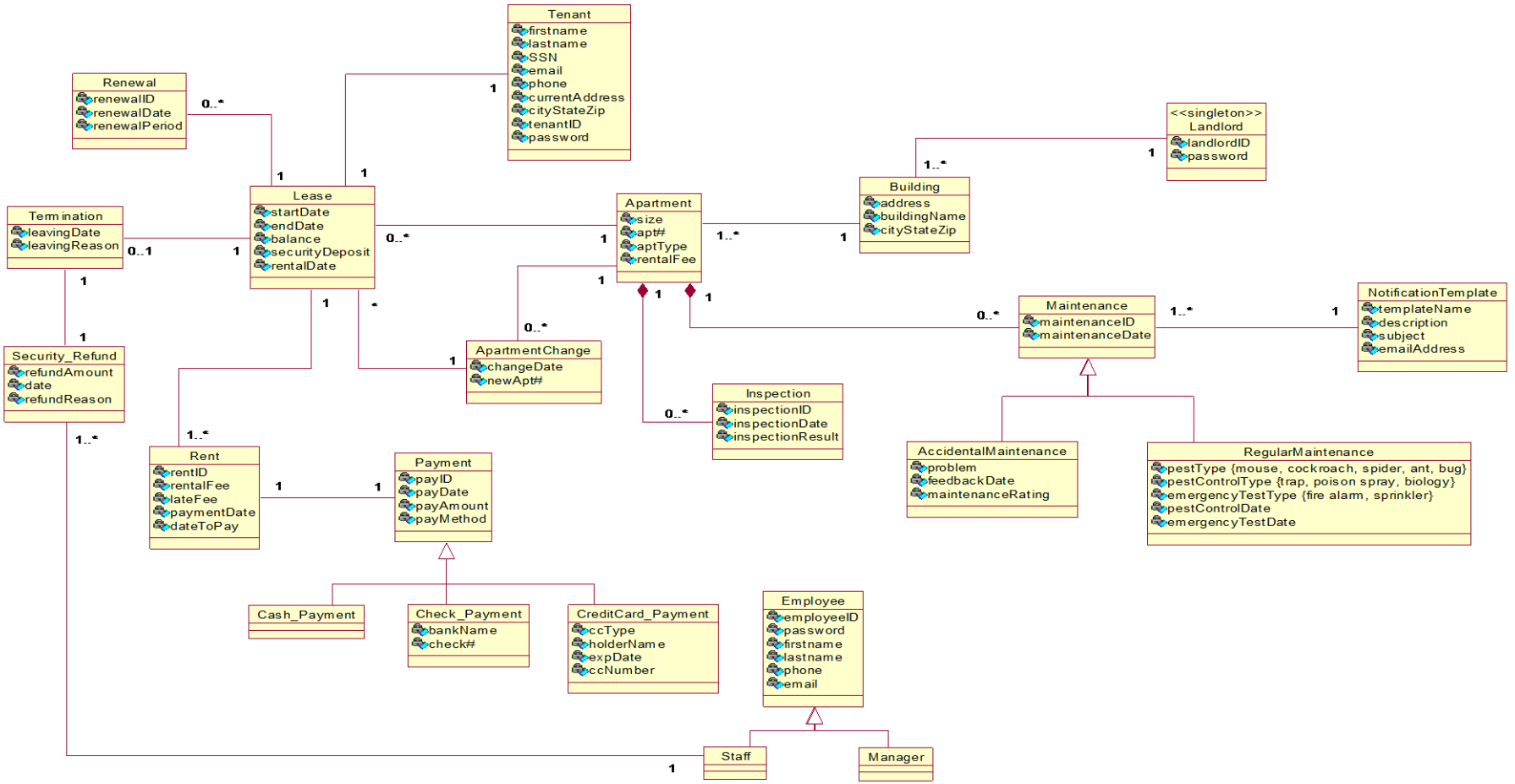
OTHER SUCCESSFUL SCENARIOS (Specify any ***successful*** variations of the *normal* execution path, including extension points using EXTEND *eus_name*)	**Step**	**Branching Action**
	1a. Staff, being asked by tenant to terminate lease, begins termination process.	2a. Document management system retrieves current lease information
	3a. Staff inputs reasons for termination of lease.	4a. System confirms that tenant is in good standing, and meets all qualifications to terminate a lease.
UNSUCCESSFUL SCENARIOS (*erroneous* situations*)*	**Conditions**	**Actions**
	4. Document Management System does not have documented proof that will allow a lease termination.	System requests more evidence, to allow a lawful eviction of tenant. Action Aborted.
	4a. Tenant is not able to terminate lease due to various reasons such as back-rent owed or legal contract in place.	System requests tenant become good-standing. Action Aborted.
Priority in scheduling	First	
Frequency	Once, at end of a tenants lease, before leaving the unit.	
Other non-functional requirements	Must be able to provide legal records, incase its needed in court in case of evictions. Must hold records in case of tenants return or request for references.	
Business rules and data logic	Tenant must be listed as primary lessee in document management system.	
Superordinates		
Developer	Nathan Vasserman	
Creation date and last modified date	1.0	
Other Comments		

USE CASE #		
USE CASE Name	Process Tenant Registration	
ACTOR	Staff, Document Management System	
Goal (1 phrase)	To register a new tenant's lease	
Overview and scope	A staff member or a Tenant will complete a registration process to register a new Tenant who will be rented an apartment	
Level	Primary	
Preconditions	Staff must be logged on.	
Postconditions in words(write in passive and past tense)	The system was registered a new Tenant. All the tenant data was recorded in the System. The tenant's amount to pay was successfully calculated. All the payment process was completed and its data recorded. A new contract was generated. The tenant's apartment was set as not available.	
Trigger	Staff selects rent a new apartment option.	
Included Use Cases	"Process credit card payment"	
Extending Use Cases		
***MAIN SUCCESSFUL SCENARIO* in numbered sequence** Reference "included use cases" in this section using INCLUDE *ius_name*	**Actor Action**	**System Action**
	1. Staff selects rent a new apartment option.	2. The System shows all the available apartments.
	3. User selects an apartment, and the date to enter to live.	4. System displays a tenant data form.
	5. User enters his first and last name, SSN, email, contact phone, contact address.	6. System checks the form data.

		7. System gets the apartment rental cost, security deposit and calculates the prepaid rent.
		8. System shows a form with all the 7. Step concepts and the payment way option (if a logged user as staff, if not, credit card is the only option).
	9. Staff selects credit card option and enters the Credit Card Type, CCName, CCNumber and CCExpiration date.	10. System performs "Process credit card payment" use case.
		11. System records the user data and payment process data.
		12. System sends tenant data to Document Manager System
		13. System records reports to the user the operation was successful, and shows the contract ready to be printed two copies with instructions to send one back by mail, fax or in person.
OTHER SUCCESSFUL SCENARIOS (Specify any ***successful*** variations of the *normal* execution path, including extension points using EXTEND *eus_name*)	**Step**	**Branching Action**
	10a.CCSystem doesn't process the payment because of a problem with the credit card.	10a. The step 9 is repeated again until the payment is process. The use case continues in step 11.
	8a. Staff selects check pay options and enters the check number and check amount.	4a. The use case continues in step 11.
	12a Document Manager System is not available	The use case finishes.
UNSUCCESSFUL SCENARIOS (*erroneous* situations*)*	**Conditions**	**Actions**
	2. There are no apartments available.	System shows a message and use case finishes.
	3a, 5a, 9a. Staff cancels the process.	Use case finishes.

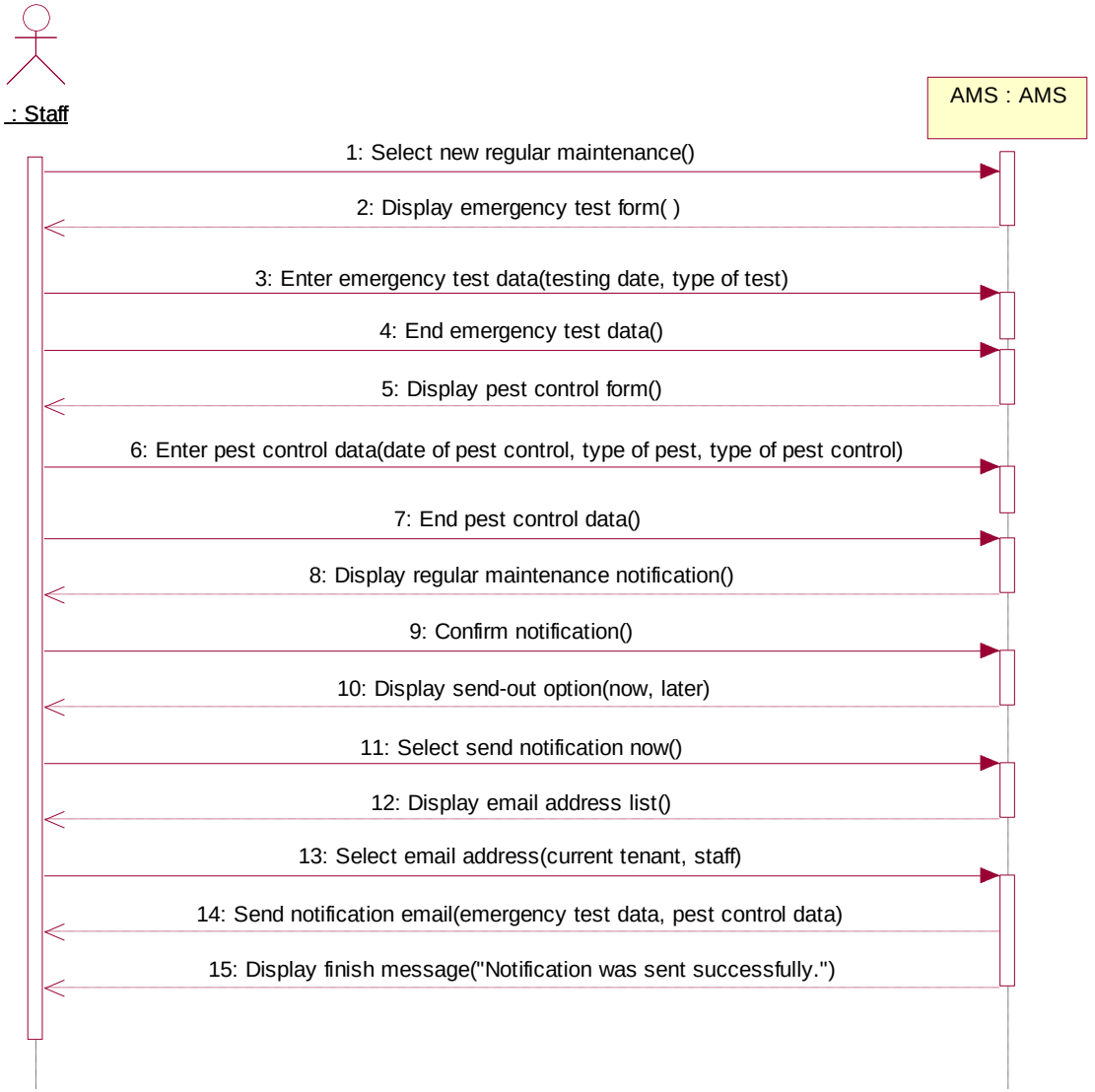
Priority in scheduling	First	
Frequency	Less than the amount of apartments per year.	
Other non-functional requirements	Connection must be secure. The contract must be shown in less than 10 seconds. User can cancel the process in every moment. Every user interface is displayed in less than 5 seconds.	
Business rules and data logic	The staff prints the two copies of the contract. Tenant must sign both. Tenant must show a valid and official ID card to staff.	
Superordinates		
Developer	David Fernandez	
Creation date and last modified date	5/10/2010	
Other Comments		

Class Diagram

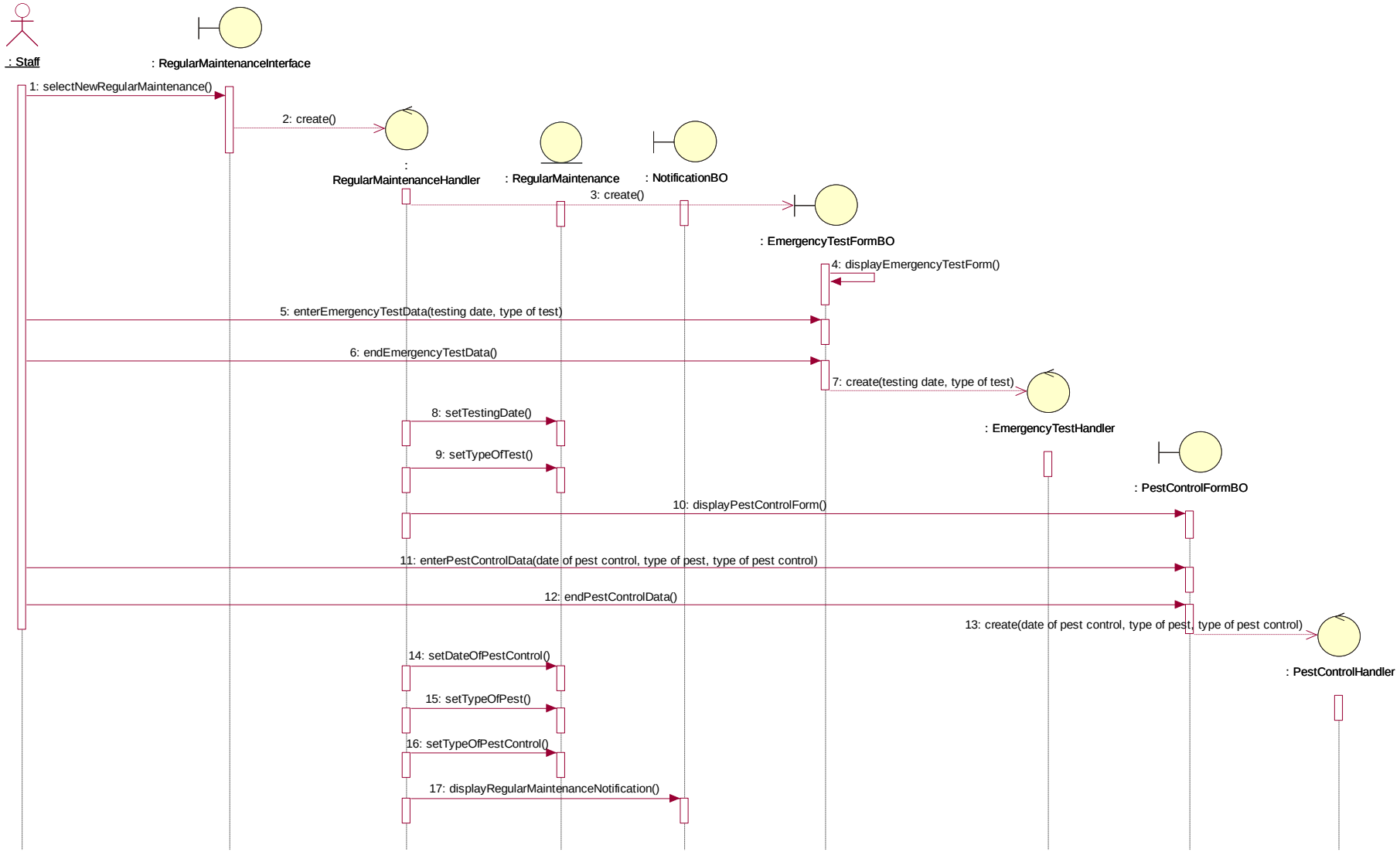


Sequence Diagrams

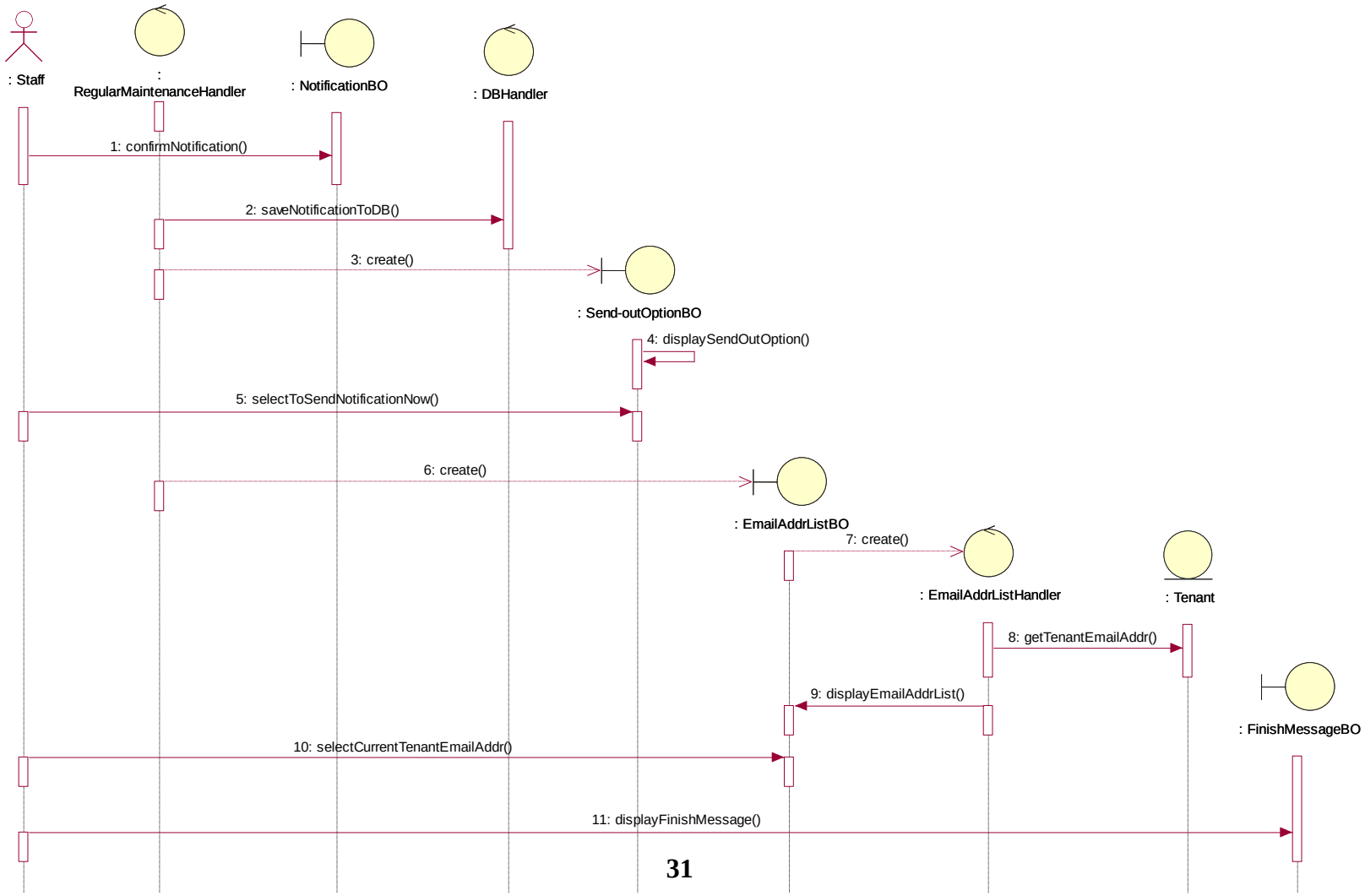
Fangwu Wei, System Sequence Diagram (Record regular maintenance)



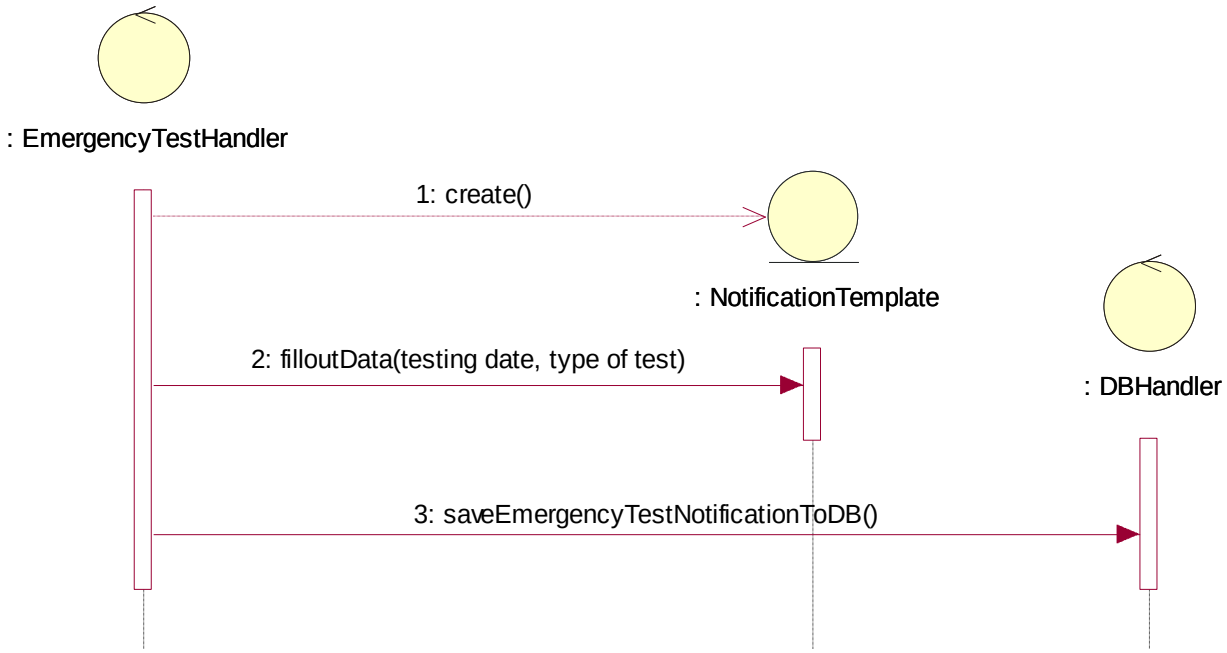
SQD 1 (Record regular maintenance)



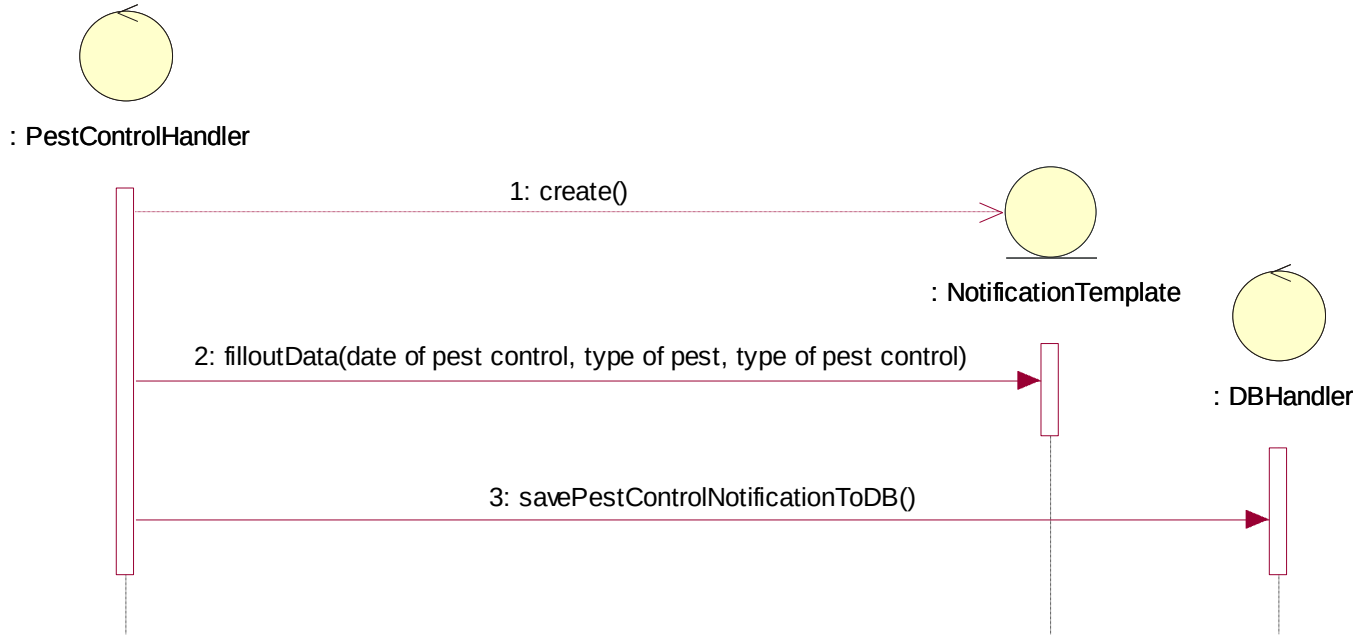
SQD 2 (Record regular maintenance)



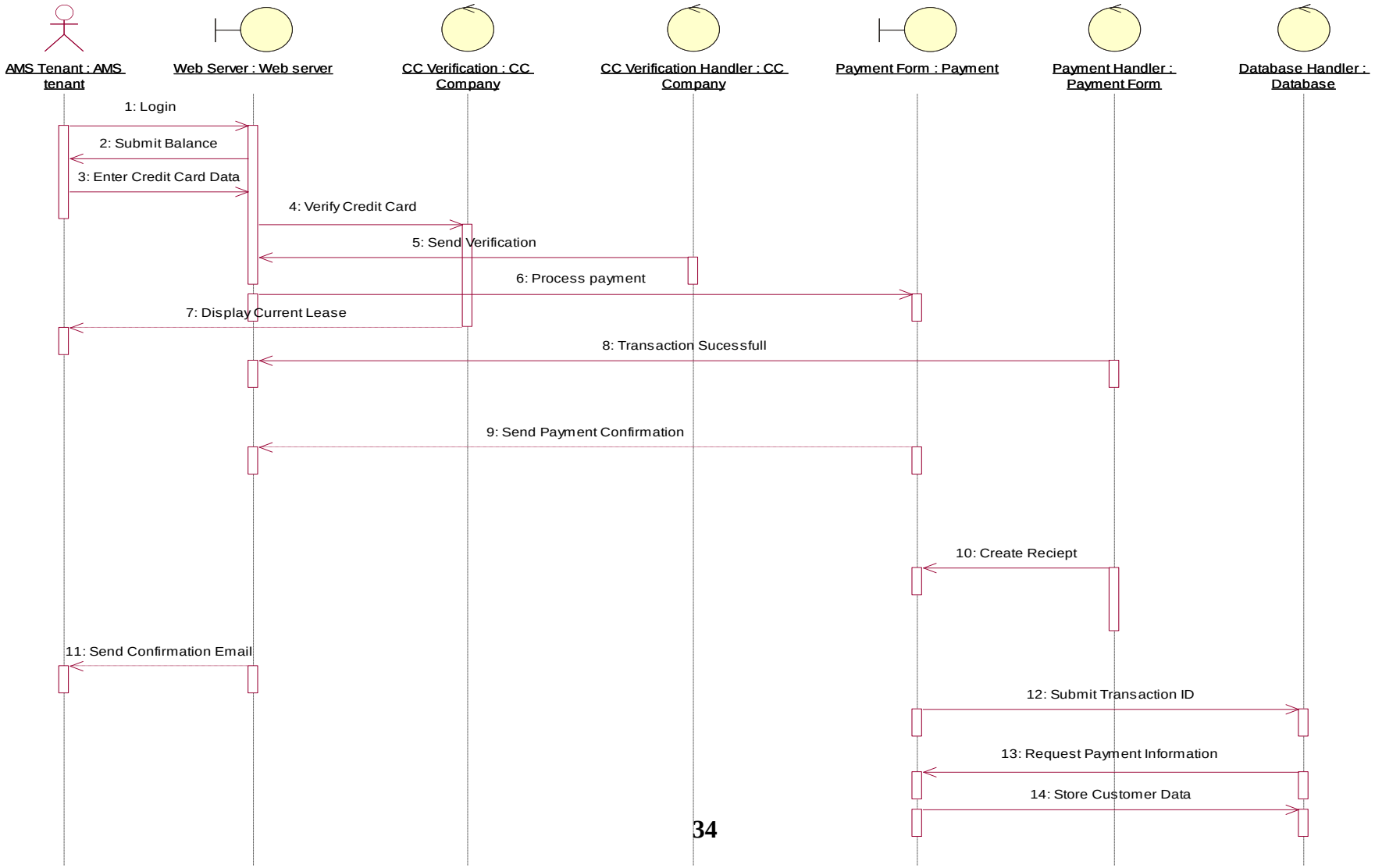
Process Emergency Test (included use case)



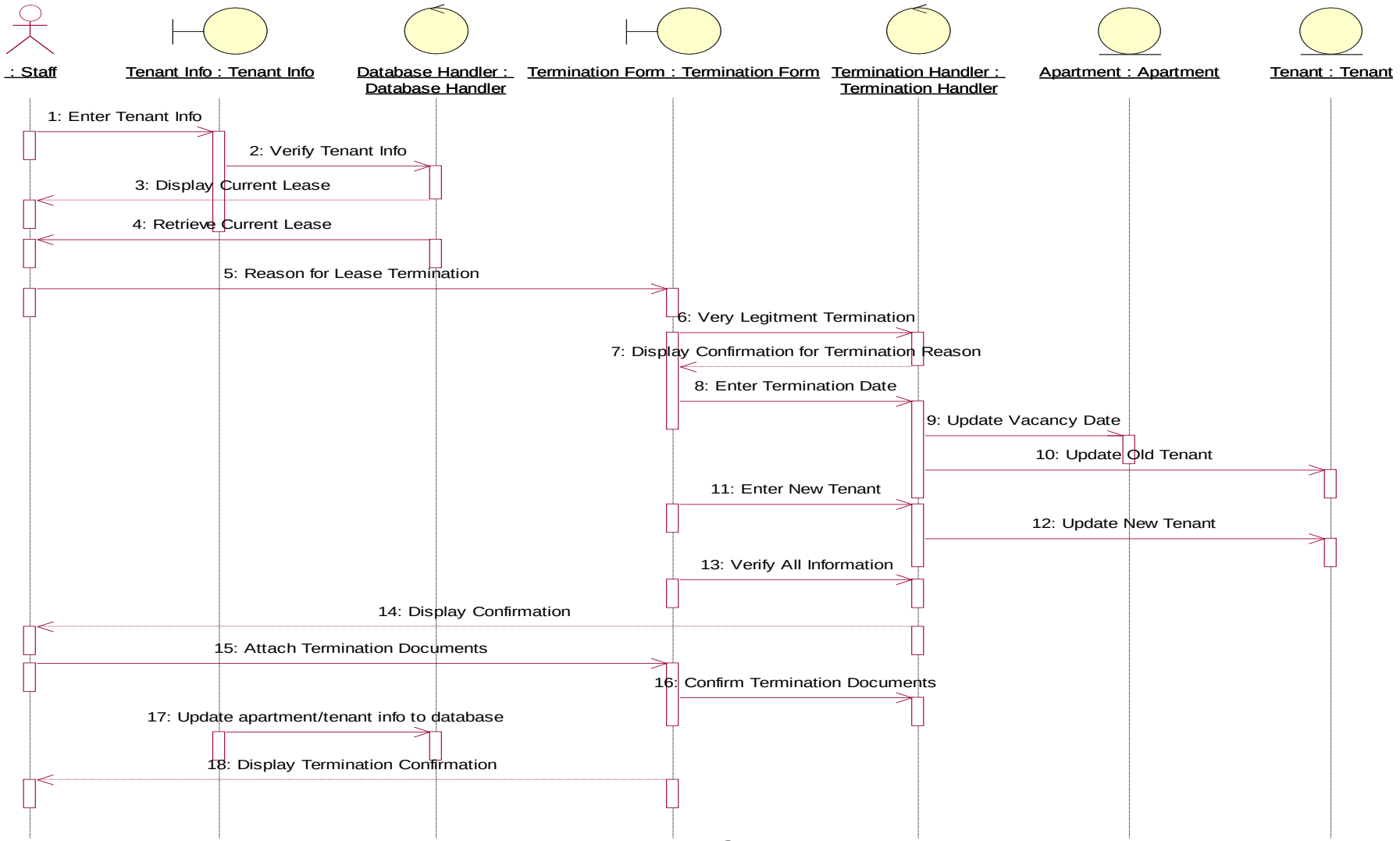
Process Pest Control (included use case)



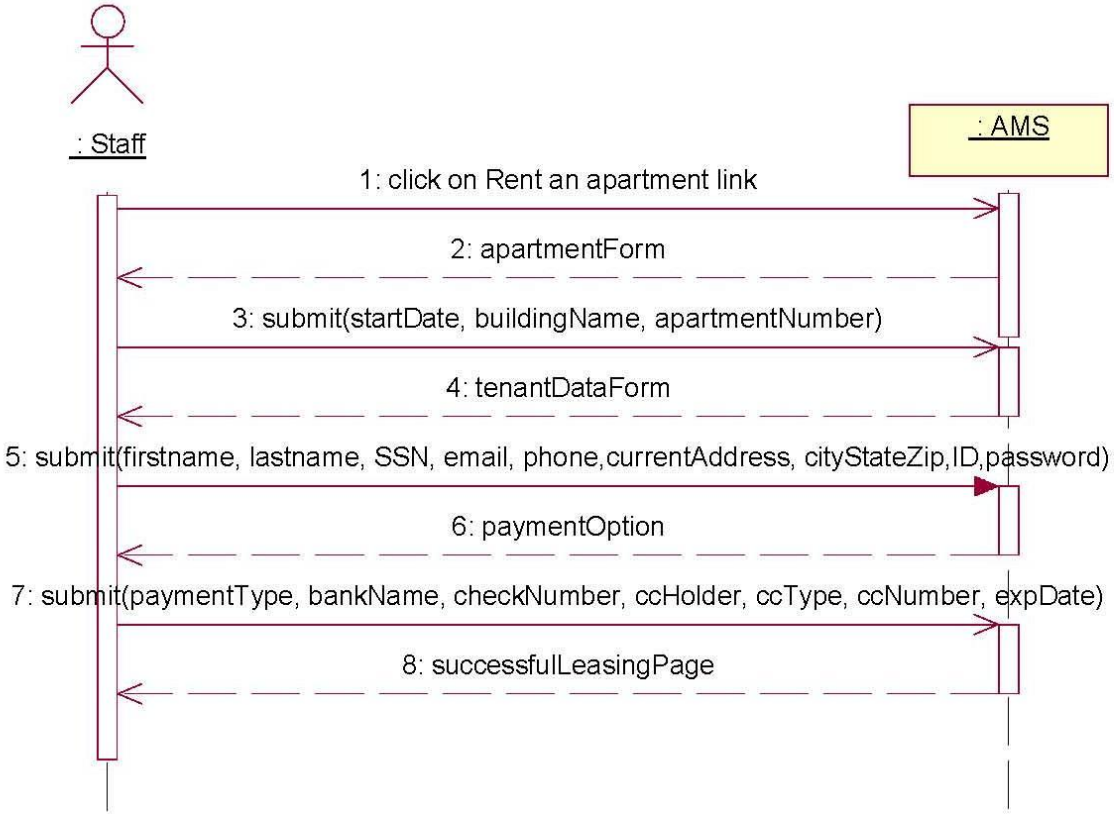
Andrew Messina, Pay Rent

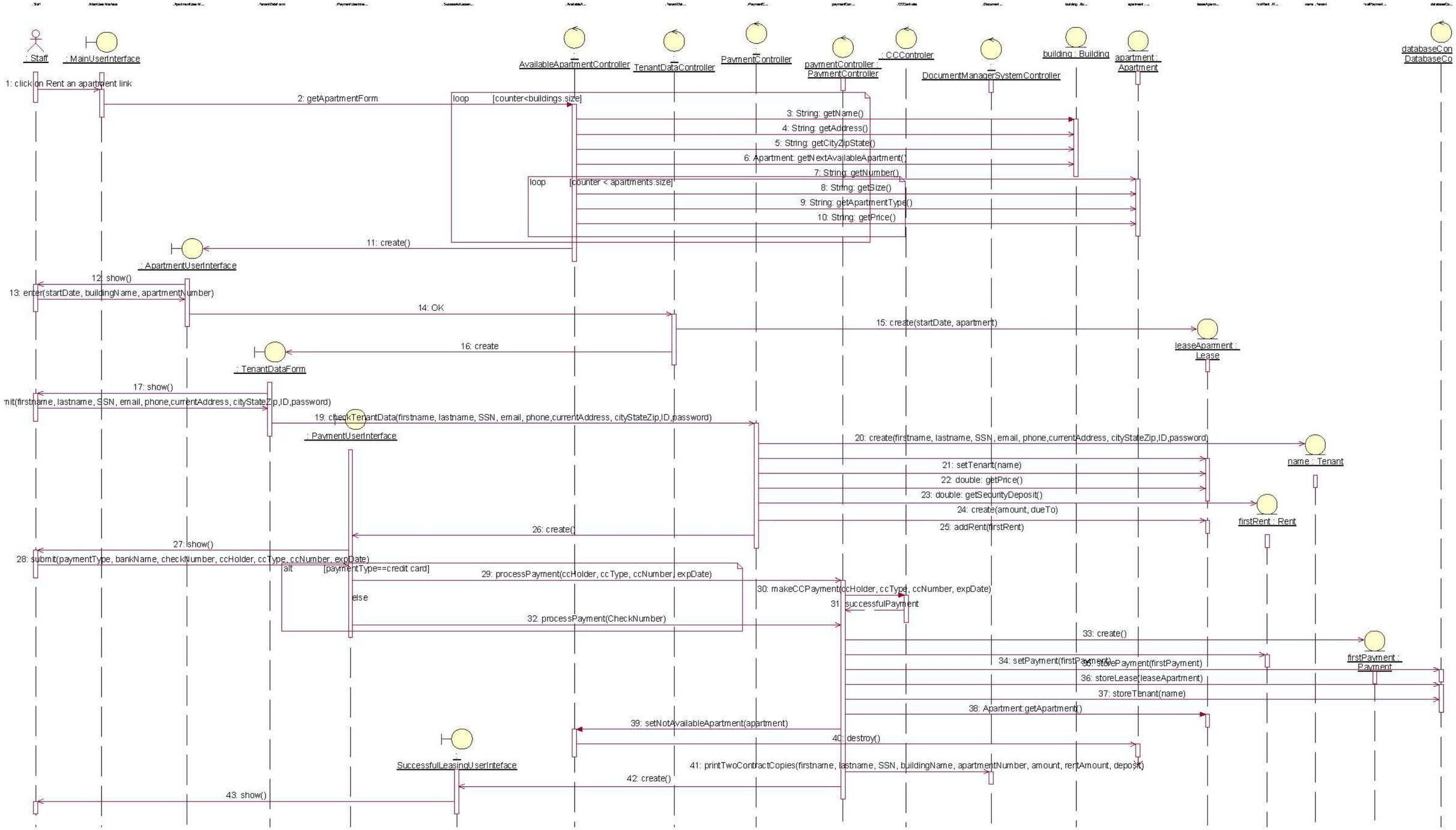


Nathan Vasserman, Terminate Lease



David Fernandez, System Sequence Diagram: Process Tenant Registration

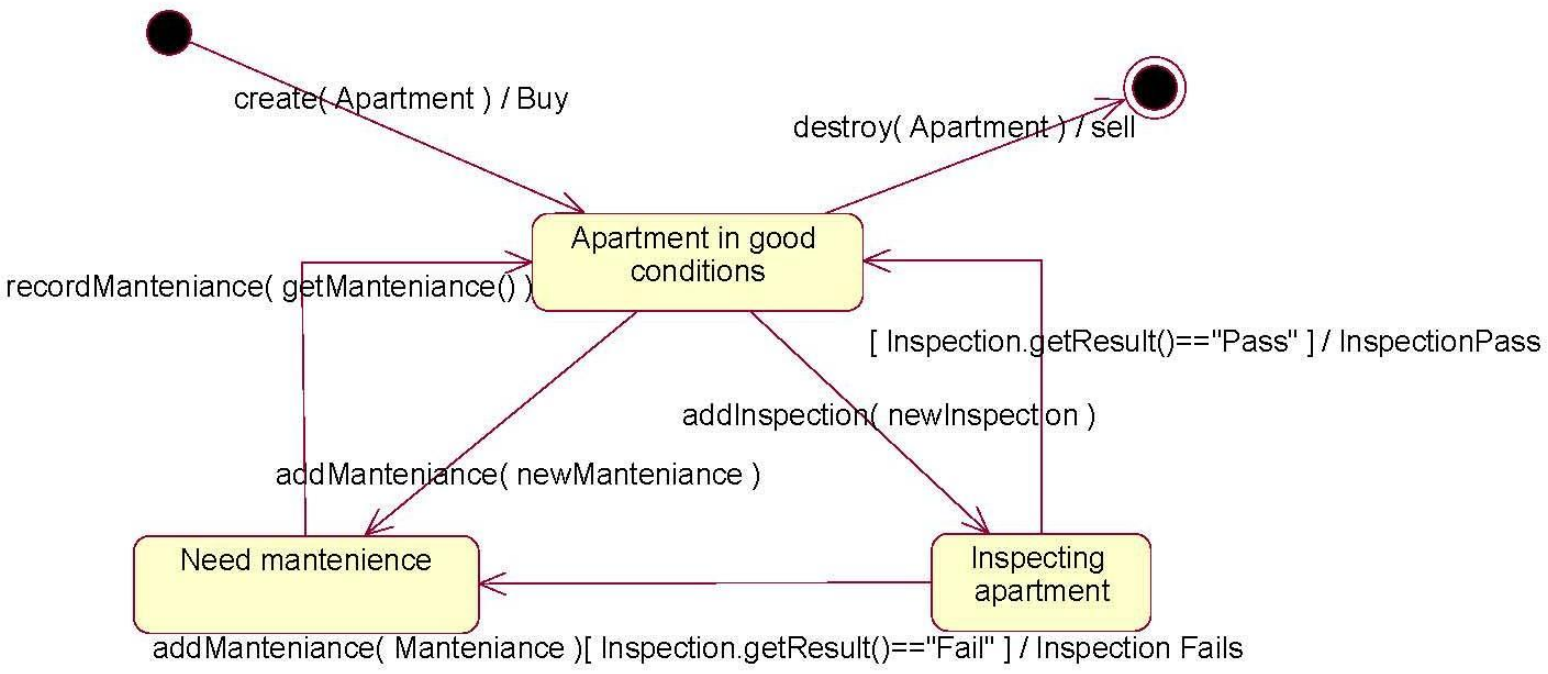



: Staff
: MainUserInterface
AvailableApartmentController
TenantDataController
PaymentController
paymentController : PaymentController
: CCController
DocumentManagerSystemController
building : Building
apartment : Apartment
databaseCon DatabaseCo
1: click on Rent an apartment link
2: getApartmentForm
loop [counter<buildings.size]
3: String: getName()
4: String: getAddress()
5: String: getCityZipState()
6: Apartment: getNextAvailableApartment()
7: String: getNumber()
loop [counter < apartments.size]
8: String: getSize()
9: String: getApartmentType()
10: String: getPrice()
11: create()
: ApartmentUserInterface
12: show()
13: enter(startDate, buildingName, apartmentNumber)
14: OK
15: create(startDate, apartment)
leaseAparment : Lease
16: create
: TenantDataForm
17: show()
nit(firstname, lastname, SSN, email, phone,currentAddress, cityStateZip,ID,password)
19: checkTenantData(firstname, lastname, SSN, email, phone,currentAddress, cityStateZip,ID,password)
: PaymentUserInterface
20: create(firstname, lastname, SSN, email, phone,currentAddress, cityStateZip,ID,password)
name : Tenant
21: setTenant(name)
22: double: getPrice()
23: double: getSecurityDeposit()
24: create(amount, dueTo)
firstRent : Rent
25: addRent(firstRent)
26: create()
27: show()
28: submit(paymentType, bankName, checkNumber, ccHolder, ccType, ccNumber, expDate)
alt [paymentType==credit card]
29: processPayment(ccHolder, ccType, ccNumber, expDate)
30: makeCCPayment(ccHolder, ccType, ccNumber, expDate)
31: successfulPayment
else
32: processPayment(CheckNumber)
33: create()
firstPayment : Payment
34: setPayment(firstPayment)
35: storePayment(firstPayment)
36: storeLease(leaseApartment)
37: storeTenant(name)
38: Apartment:getApartment()
39: setNotAvailableApartment(apartment)
40: destroy()
SuccessfulLeasingUserInteface
41: printTwoContractCopies(firstname, lastname, SSN, buildingName, apartmentNumber, amount, rentAmount, deposit)
42: create()
43: show()

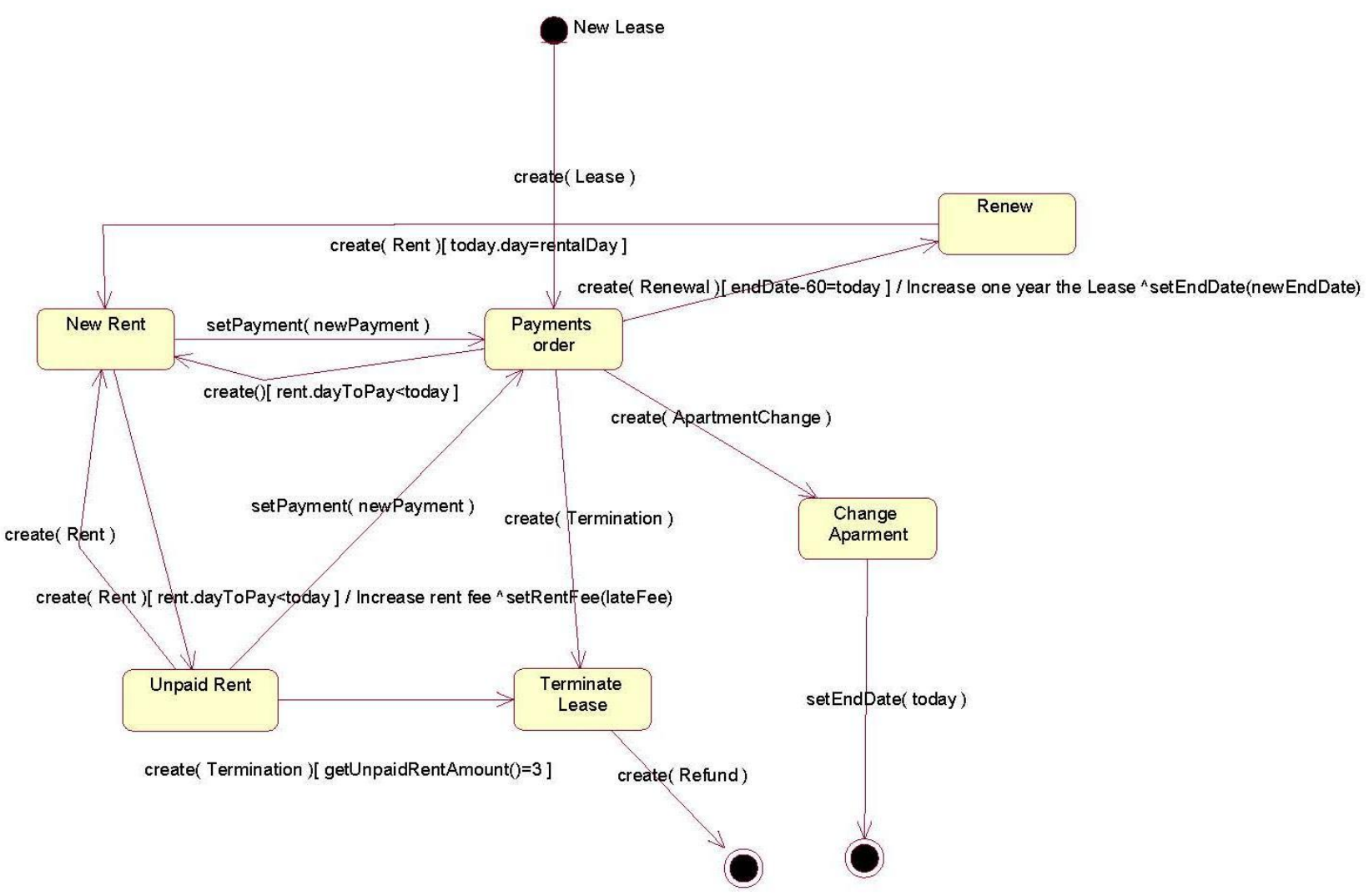
State Diagrams for Apartment and Lease objects

Apartment and Lease are the most relevant classes of Domain class diagram.

Apartment

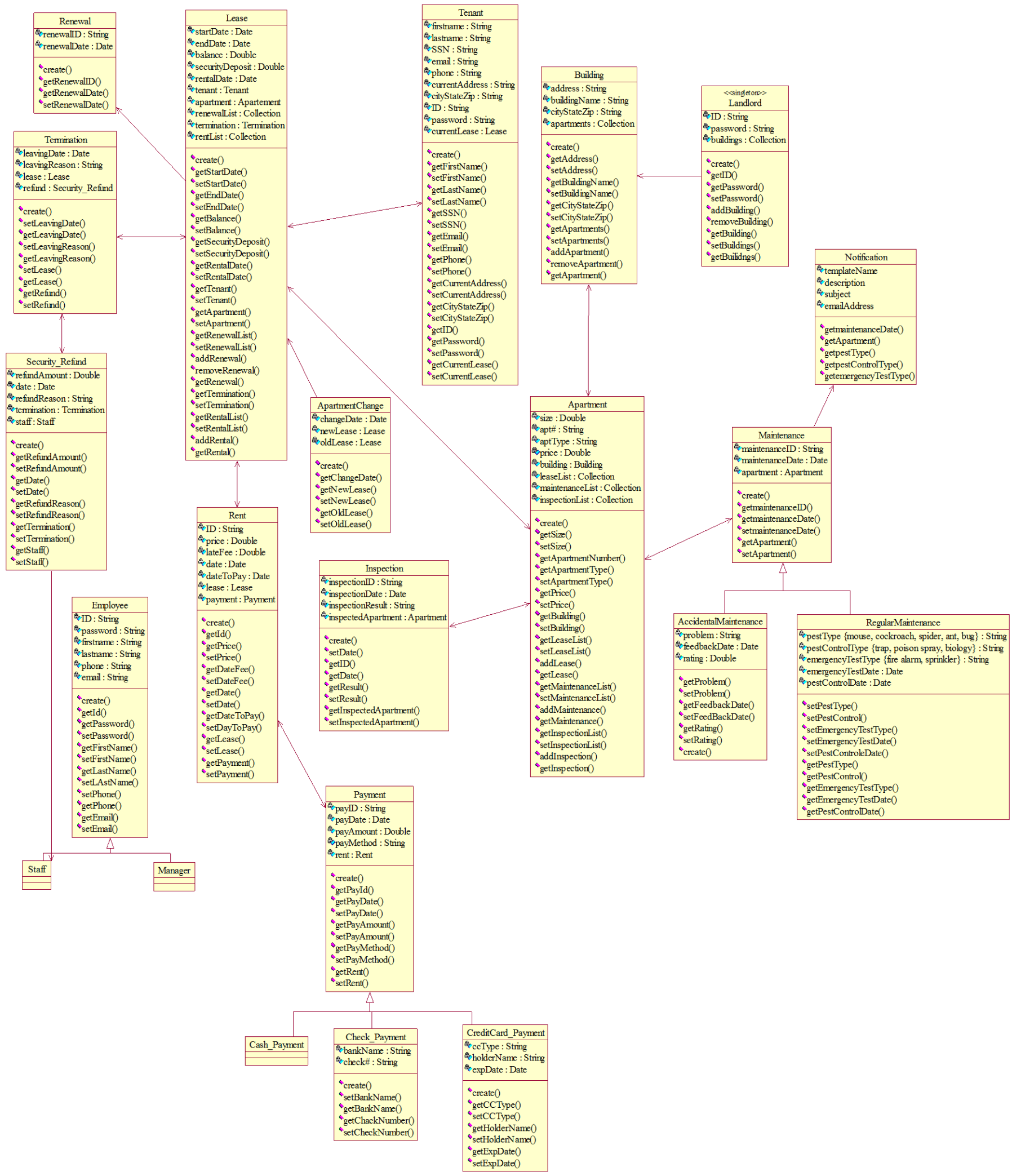


Lease

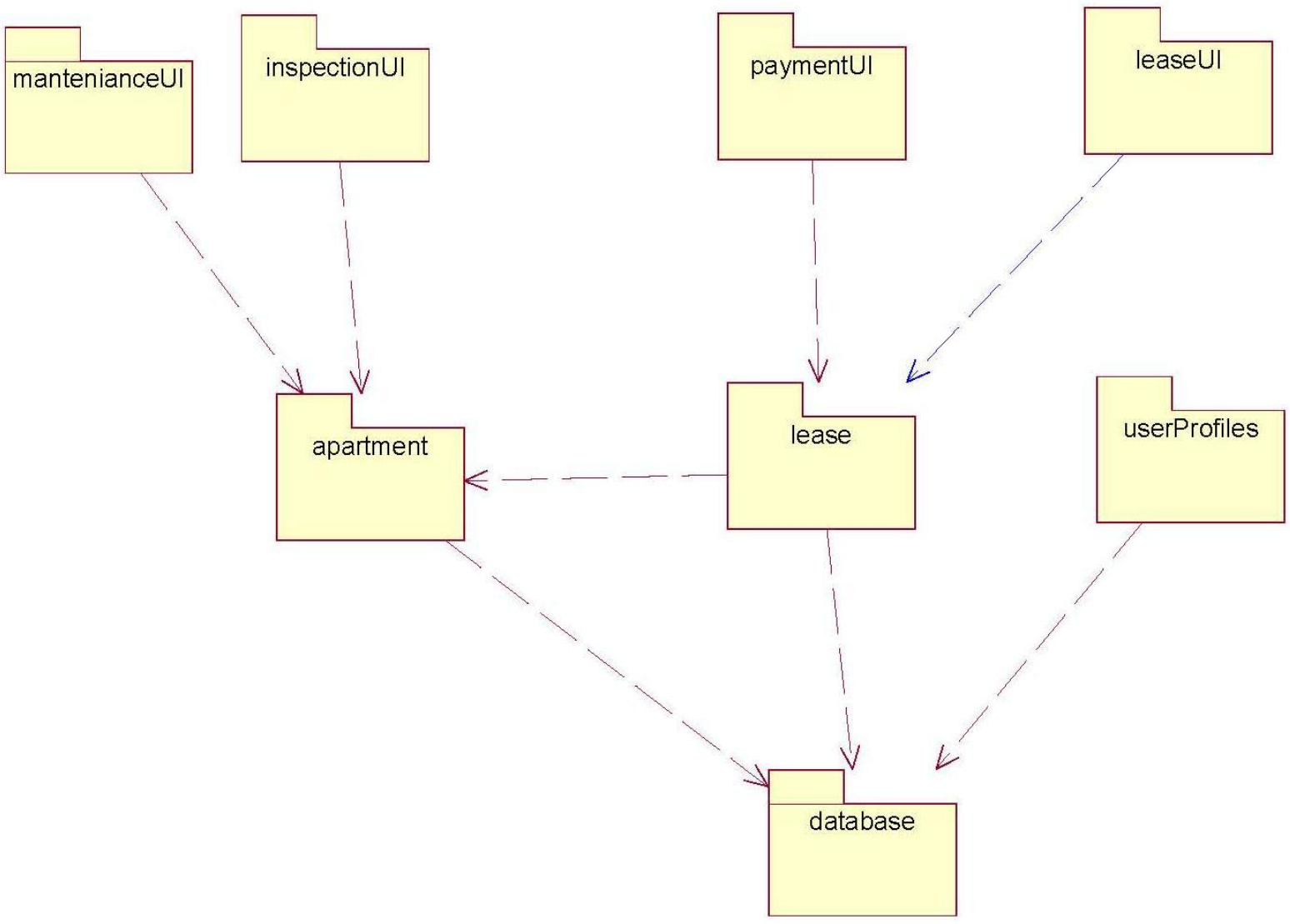


Fangwu Wei, Andrew Messina, David Fernandez Galende, Nathan Vasserman
Group Project Submission 6/10/2010

Design Class Diagram



Package Diagram



MaintenanceUI package contains all screen and controllers classes used in the record regular maintenance sequence diagram.

PaymentUI package contains all screen and controller classes used in Pay Rent sequence diagram.

LeaseUI package contains all screen and controller classes used in Terminate lease and Process tenant registration sequence diagrams.

Database package contains all the controllers in the all sequence diagrams which interact with the database.

User profile package contains the classes: Tenant, Employee, Staff, Manager, and Landlord.

Lease package contains: Lease, Rent, Payment, CreditCard_Payment, Check_Payment, Cash_Payment, ApartmentChange, Termination, SecurityRefund and Renewal.

Apartment package contains: Apartment, Building, Inspection, Maintenance, AccidentalMaintenance and RegularMaintenance.

Database schema

Database schema

Landlord (landlordID, password)

Employee (employeeID, password, firstname, lastname, phone, email)

Staff (employeeID*)

Manager (employeeID*)

Tenant (tenantID*, password, SSN, firstname, lastname, email, phone, currentAddress, cityStateZip, password)

Lease (leaseID, startDate, endDate, balance, securityDeposit, rentalDate, tenatId*, apartmentNumber*, terminationID*, apartmentChangeID*)

Building (buildingName, address, cityStateZip, landlordID*)

Apartment (apartmentNumber, size, aptType, rentalFee, buildingName*)

ApartmentChange(apartmentNumber, changeDate, newAptNumber, apartmentNumber*)

Renewal (renewalID, renewalDate, renewalPeriod, leaseID*)

Termination(terminationID, leavingDate, leavingReason ,)

Security_Refund(SRID, refundAmount, date, refundReason, terminationID*, employeeID*)

Rent (rentID, rentalFee, lateFee, date, dateToPay, leaseID*, payID*)

Payment (payID, payDate, payAmount, payMethod, rentID*)

Cash_Payment (payID*)

Check_Payment(payID*, bankName, checkNumber)

CreditCard_Payment(payID*, holderName, expDate, ccType, ccNumber)

Inspection (apartmentNumber*, inspectionID, inspectionDate, inspectionResult)

Maintenance (apartmentNumber*, maintenanceID, maintenanceDate, templateName*)

AccidentalMaintenance ((apartmentNumber*, maintenanceID)*, problem, feedbackDate, maintenanceRating)

RegularMaintenance ((apartmentNumber*, maintenanceID)*, pestType, pestControlType, emergencyType, pestControlDate, emergencyTestDate)

NotificationTemplate (templateName, description, subject, emailAddress)

Discussion on developing the diagrams.

Our class diagram was developed with all four group members communicating via Skype. There was much debate over the existence of certain classes. One of the biggest discussions within the group was regarding how lease terminations should be handled. We came to the consensus to form a termination class with relationships to a security deposit refund, the lease class and branching options for renewal. Maintenance was ultimately decided to be split into 2 generalizations because of the nature of different types of maintenance and its relation to each apartment unit. We also debated between what constitutes an “employee” of the building staff. This was decided to be a single class that could encapsulate a repairman, clerical staff or assistant property manager. The handling of the employee class is on the administrative side of the building. Landlord was a separate class altogether, even though his involvement with the apartment management system is minimal.

Our use case diagram was a similar approach as well. Since the use case descriptions and sequence diagrams were an individual effort, we had to each choose our own use cases, and incorporate them into a much large diagram that would involve several use cases we knew would not be addressed as in-depth as others. Our use case diagram ending up involving many more actors then our class diagram would appear to show. In this instance we had to separate out many of the roles that the apartment staff would manage, in addition to 3rd party inspectors, landlords, managers and the payment management systems. Most of the use case diagram centers around the steps involved in simply paying rent and keeping the apartment database and their tenants up to date. The group collaborative effort allowed us to share many of our experiences, especially those of us with first hand experience living in an apartment building compared to those who haven’t.

Summary and Lessons Learned

After the completion of this project, we felt we have learned a significant amount about the work and processes that go into handling apartment management. There are many different players in the process, many of whom we’ve never actively thought about. This includes not only the Landlord and Tenant, but also apartment staff, supervisors, employees and where they all fit in to managing an apartment unit. Our class diagram design involved a lot of collaborative thinking to figure out where they players fit and what roles do they accomplish.

The job of creating a design document for the system taught us much in how UML is structured. There are a lot of aspects of it that are still confusing such as some of the different objects in a sequence diagram. Thinking about the sequence diagram and the class diagram, I would say these were our biggest challenges. Some of the aspects of the system, such as paying rent, are very complicated and required a lot of critical thinking on our parts to be able to identify all the different scenarios and sequence steps for proper identification. The project in general definitely did a lot to expand our skills in UML and modeling and hopefully we will be able to use these skills in the future.

Appendix

Division of Work

Problem Statement: Group Effort
Class Diagram: Group Effort
David: Process Tenant Registration use case description and sequence Diagram, Database Schema
Andrew: Process payment/rent use case description and sequence Diagram
Nathan: Terminate Lease use case description and sequence Diagram
Fangwu: Notify for regular maintenance use case description and sequence Diagram
Document Compilation: Nathan, Fangwu

Lessons Learned

David

This Project has consolidated my knowledge in two areas: Object Oriented Analysis and Design applying UML and team working.

In OOA/D, we have got to implement INFO620 lectures knowledge in OOA/D into a medium-large complex project as AMS. Under "if I do it, I understand it" philosophy, I understood much better the OOA/D process applying UML, translating these concept into AMS project by UML diagrams: use cases, class diagram, sequence diagram, design class diagrams and database schema.

I have learned how to develop use cases from the domain problem, identifying actors and their goals, how to do a detail description of use cases developing one of the most complex one: defining pre-conditions and post conditions for "Process tenant registration", defining the logical steps in the main scenario, some alternative (un)successful scenarios, and including non functional requirements and business rules.

I have learned to design the static vision of the system developing the domain class diagram, the dynamic vision developing the sequence diagram for "Process tenant registration", and to join both visions into the design class diagram.

I have learned how to apply the concepts to identify a good candidate class to AMS ones, its attributes, and its associations which has ever been the most difficult part for me.

I have learned how to identify and how to use controller classes and how to develop a sequence diagram from the use case description. I have started to distinguish between domain classes (class diagram), which support the business logic for AMS (Apartment, Lease, Termination, Rental, etc) from the architecture and design classes which organizes the system structure (DatabaseController, TenantDataController, …), making independent domain classes from system interfaces as user, database, other system.

I have learned how to check the consistency between diagrams: all the classes in class diagram must participate at least in one use case. In all interactions in sequence diagrams between domain objects, whose classes must exist in the class diagram, must have an association in the class diagram (unless it was too obvious) and all the messages between domain classes must be transform into class operations.

Working in groups has taught me the difficulty to coordinate and meet in a 4 member group, but how rich are the diagram when pick up all the ideas of every member. I could not develop a use case diagram with so many use cases by myself. Putting together the best ideas of every team member has allowed identifying 30 use cases!!!

Other positive experience has been to develop the class diagram. We did it altogether. Although it took us a lot of time, the result is a class diagram which met the use cases with only a little corrections from the professor. I could not have developed the class diagram without correction from the professor.

I still have a lot of thing to learn in this area of OOA/D. For example, to identify and apply design patterns. With the exception of View-Model-Controller pattern, I have not applied any pattern in AMS project.

I have no very clear how to go further than design class and package diagram. It is not very clear how to develop component and deploy diagrams.

Andrew

Coming into this class, I had a decent knowledge of UNL and diagrams. In my previous MIS classes I learned about all the diagrams we learned in this class. I also used Rational Rose, so I was somewhat prepared for this class. I still found this class extremely difficult, even though I had prior knowledge. I am a very messy person, so developing diagrams is very hard for me. Although the class was challenging, I still feel that I learned a lot about UML and the relevant diagrams. I feel that I will confidently be able to apply this knowledge and the diagrams I have learned in the future. This project also pushed me to spend a lot of time developing a sequence and class diagram, which are my two least favorite. I am glad I received a better understanding of both diagrams though this class.

Fangwu Wei, Andrew Messina, David Fernandez Galende, Nathan Vasserman
Group Project Submission 6/10/2010

Nathan

I felt I learned a lot the planning and preparation that goes into software design. Since my division was in the terminate lease segment, there was a significant amount of research that needed to be done into rental laws in Pennsylvania. Making sure all those laws were compliant with our Apartment Management System. As of writing this, I do not know what kind of grade I will receive on my sequence diagram, but I am hoping it is acceptable as I put it in a significant amount of time and effort into it. Going through the logical steps of software design sequences was a bit daunting for me and I think I learned a lot about it.

Fangwu

- Learned how to create UML diagrams, including use case diagram, class diagram, and sequence diagram.
- Learned how to identify the use case, class, object, and then draw the diagrams based on rules, notation, and heuristic evaluation steps.
- Learned how to analyze and design a system in a real world, and develop it further based on the UML diagrams, database, and architecture.

Unanswered Questions

1. How to identify the control object and association between control object and boundary object/entity object? I felt better after I read the paper, “Developing Sequence Diagrams in UML”, but I was not so clear about control object. I didn’t create a control object before in my java program because of my basic java programming skills. I guess to practice programming and UML diagrams both would be better for me, which helps me be object-sensitive.

2. I learned the relational database last week. I used the ER Diagram and Data Flow Diagram (DFD) in my passed projects. Are those diagrams necessary in an OOA/D? If we need them, which step we need to use them?