Android Project Process & Products

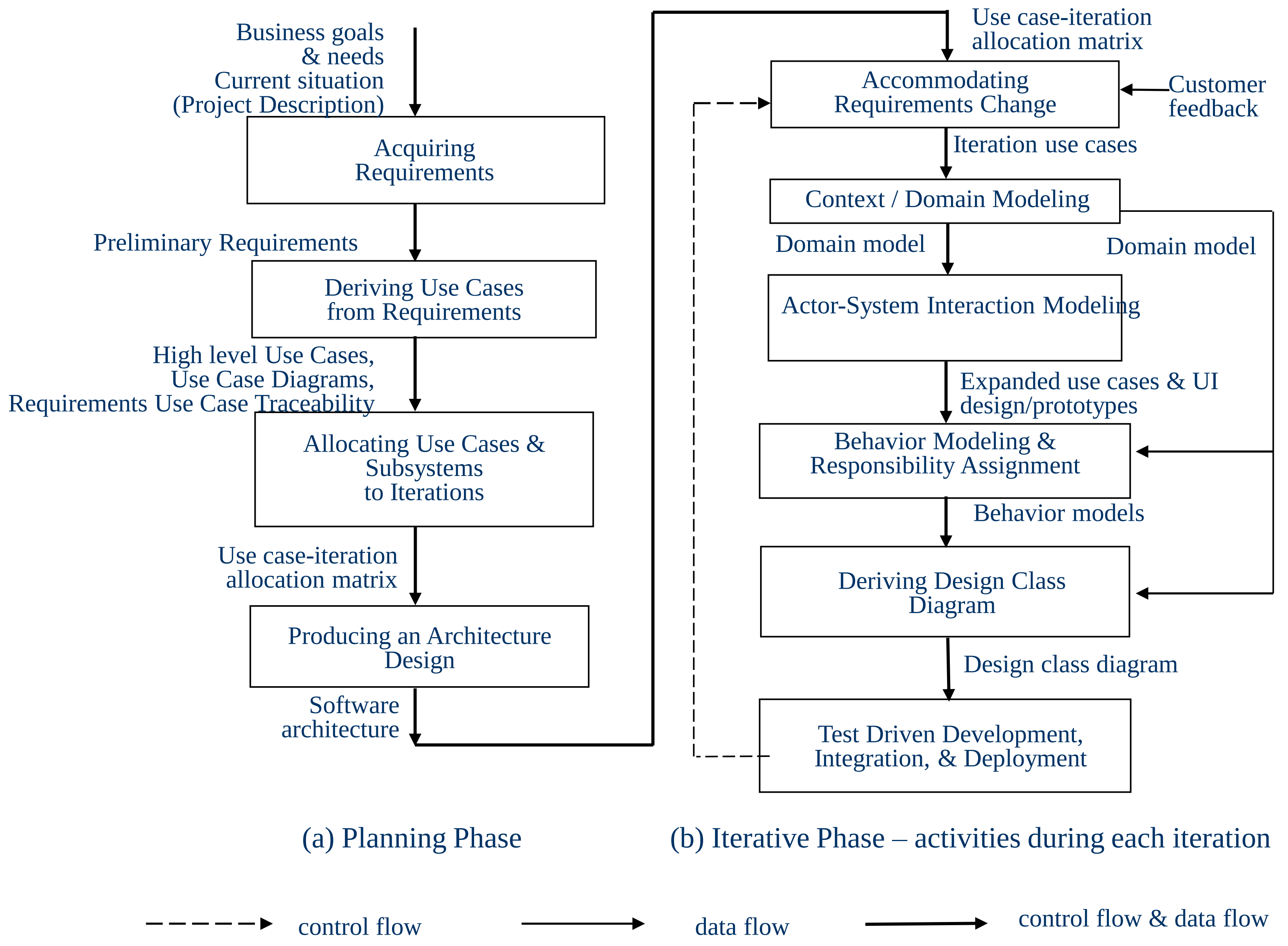
CSE 5324 Software Engineering: Analysis, Design, and Testing

Dr. Michael F. Siok, PE, ESEP
Adjunct Professor
University of Texas at Arlington

Android Agile Project Iterations

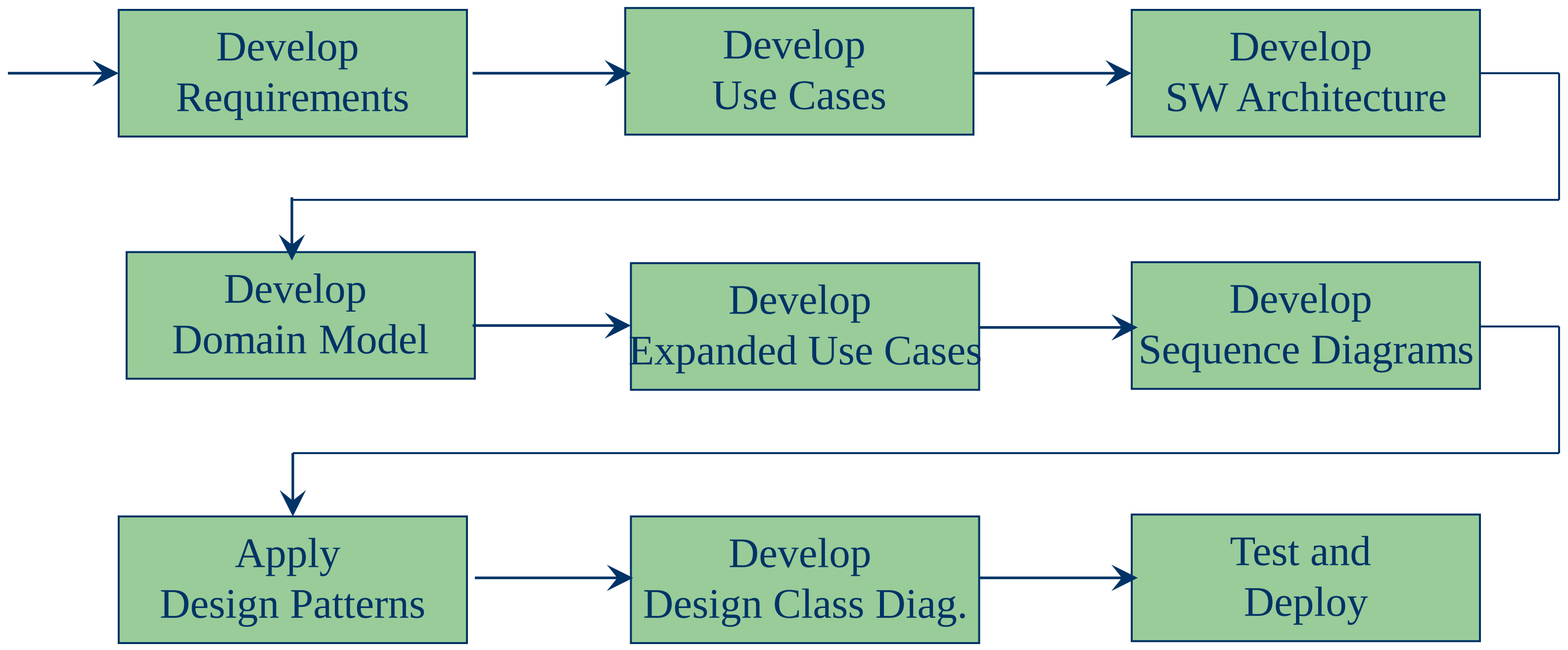
- Overview of Android project process, iterations, and work products
  - Agile Process Overview
  - Iterations and expected work products
  - Example Iteration Work Products

CSE 5324 Class Project Agile Process


Business goals & needs
Current situation (Project Description)
Acquiring Requirements
Preliminary Requirements
Deriving Use Cases from Requirements
High level Use Cases, Use Case Diagrams, Requirements Use Case Traceability
Allocating Use Cases & Subsystems to Iterations
Use case-iteration allocation matrix
Producing an Architecture Design
Software architecture
Use case-iteration allocation matrix
Accommodating Requirements Change
Customer feedback
Iteration use cases
Context / Domain Modeling
Domain model
Domain model
Actor-System Interaction Modeling
Expanded use cases & UI design/prototypes
Behavior Modeling & Responsibility Assignment
Behavior models
Deriving Design Class Diagram
Design class diagram
Test Driven Development, Integration, & Deployment
control flow
data flow
control flow & data flow

(a) Planning Phase (b) Iterative Phase – activities during each iteration

Class Agile Process (a different view)



Iteration Products

- Iteration 1:
  - Project Description
  - Requirements (spreadsheet)
  - High Level Use Cases Listing (TUCBW, TUCEW)
  - Use Case Diagrams
  - Requirements to Use Case Traceability Matrix (RUCTM)
    - Maps requirements to Use Cases – makes sure we did not miss any requirements in the iterations work
  - Use Case Iteration Matrix
    - Schedules when Use Cases are implemented/programmed
  - Domain Model
    - Developers' view of the '*problem space*'

Iteration Products

- Iteration 2:
  - Expanded Use Cases (EUC)
  - User Interface Prototypes (for each EUC)
  - Sequence diagrams
  - Any other model/drawing/requirements/UC updates

- Iteration 3:
  - Design Class Diagram (DCD)
  - Any other model/drawing/requirements/UC updates
  - Demonstration (in-class)

Example Project (parts of *UTA Housing*)
Iteration 1 Materials

(Each project iteration submittal is in PDF format,
a single file with the following format: <*TeamName*-Group*n*-It*x*.pdf>)

The project submittal represents a single design package for the team's app project.
This design package builds with each iteration.

The figures that follow are examples only to illustrate diagram expected format and content
Design items must be consistent across drawings.
For example, names of UCs must be exactly the same in UCDs,
Iteration matrix, RTUCM, EUCs, UIPs, DCD, etc.

This is the Project Notebook Title Page

App Name

Team Number and Name

Team Member Names

Date

Iteration #

Table of Contents

- ToC goes here (with page number references)

Team Project Description

-- *Goes here* –

-- line numbered, except for Team bios --

Unique requirement number needed for each requirement

Refers back to lines of the project description as the source of requirements or the requirement is derived from this source and the team's analysis.

Requirements

Req ID	Req Statement	Line reference
R1	The system shall provide for authenticated login to the Housing app	5, 6
R2	The system shall provide a reset password function for registered users	7
R3	The system shall provide for authenticated residents viewing of housing options	12 to 16
R4	The system shall provide for storage of housing floor plans and related images	21 to 23
R5	The system shall provide for viewing of housing floor plans and related images by registered users of the app	25
R6	The system shall provide for storage of feedback reviews from previous residents and other users of the app.	Derived
R7	The system shall provide for viewing of feedback reviews from previous residents and other users of the app.	35 to 38
R8	The system shall allow residents to indicate ratings for reviewed housing options	40
R9	The system shall allow authenticated users of the app to apply for housing using an on-line app submittal process applicable to each house complex.	44 to 56
R10	The system shall allow authenticated residents of the app to view the current status of their submitted application once it has been received by the housing complex.	Derived

Also known as Abstract Use Cases

Use Case List (optional, but handy)

Use Case #	Use Case Name
UC1	Login
UC2	View on Campus Housing
UC2.1	View Images and Floor Plans
UC2.2	View Reviews
UC2.3	Provide Feedback
UC 3	Apply to on-campus housing
UC 4	Check Application Status
UC 5	Reset Forgotten Passord

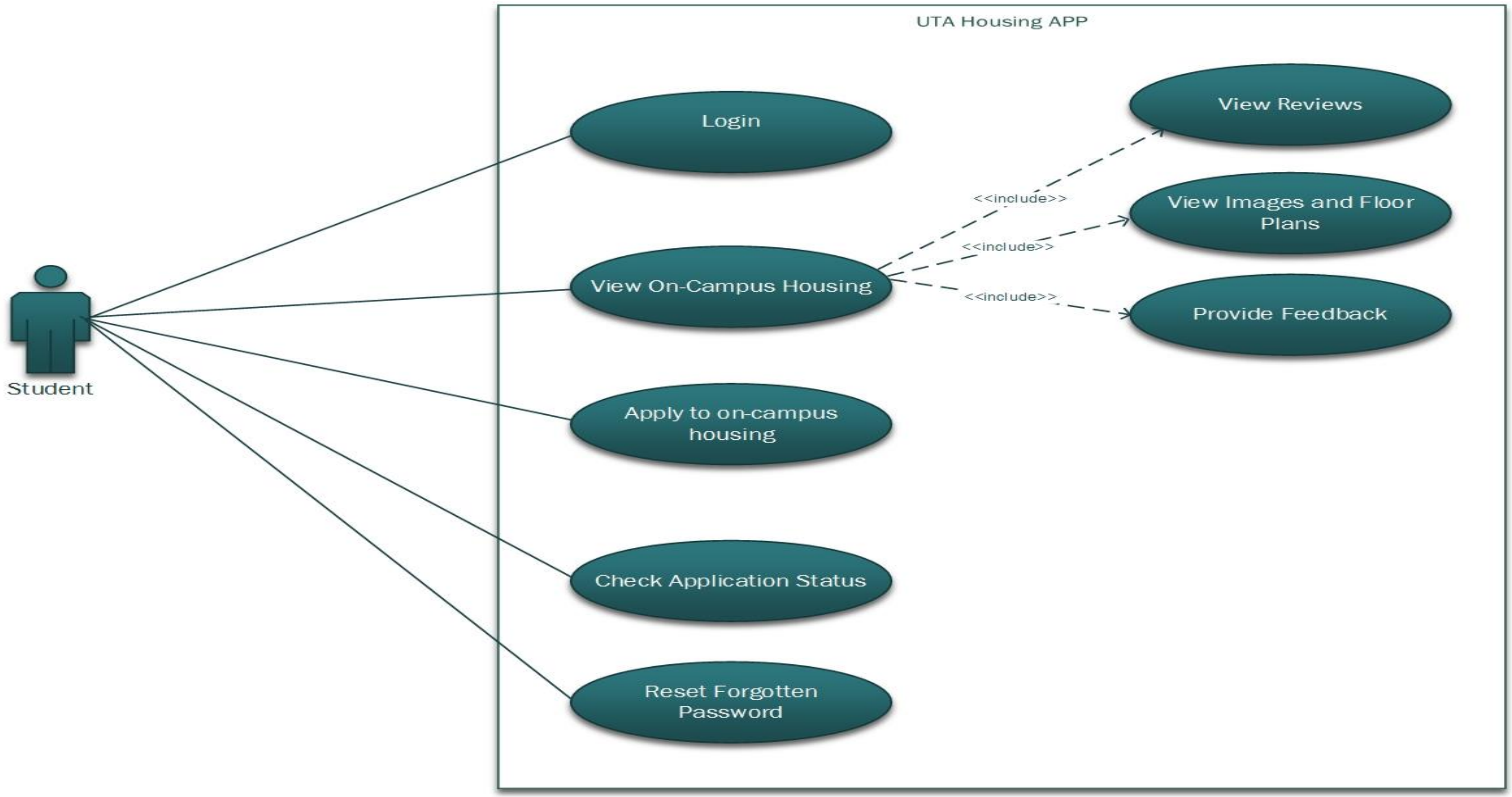
TUCBW – The use case begins with . . .
TUCEW – The use case ends with . . .

High Level Use Cases

- **UC 1: Login**
  - TUCBW the student enters his MyMav credentials, and clicks on Sign In button.
  - TUCEW student gets signed in and the ‘Housing Home’ screen is displayed.
- **UC 2: View on campus Housing**
  - TUCBW the student clicks on ‘View housing’ menu from the navigation drawer.
  - TUCEW the student completes viewing of the on-campus housing and sees the ‘Housing Home' screen.
- UC 2.1: View images and floor plans
  - TUCBW the student selects a housing unit from the pull-down menu on the ‘View Housing’ page.
  - TUCEW the student completes viewing of housing images and floor plans, and sees the ‘Housing Home' screen.
- UC 2.2: View reviews
  - TUCBW the student scrolls down to the ‘User Reviews’ section on the ‘Housing Details’ screen.
  - TUCEW the student views the reviews for the particular housing unit.
- UC 2.3: Provide feedback
  - TUCBW the student enters the feedback, selects desired star rating and clicks on ‘Submit Feedback’.
  - TUCEW the review is submitted and displayed in the list of reviews.
- **UC 3: Apply to on campus housing**
  - TUCBW the student clicks on ‘Apply’ menu from the navigation drawer.
  - TUCEW the student sees the housing application confirmation message.
- **UC 4: Check application status**
  - TUCBW the student selects an application and clicks on the ‘Check Status’ button.
  - TUCEW the app displays the waitlist status of the application to the user.
- **UC 5: Reset forgotten password**
  - TUCBW the user clicks on ‘Forgot Password’ on the login screen.
  - TUCEW the password is reset and the new password is sent to the student’s registered UTA email address.

Submittal could contain multiple UCDs. For example, the included use cases in the diagram below could be removed from this diagram and another UCD added named "View On-Campus Housing" containing these 3 use cases associated with the student actor.

Use Case Diagram



This matrix accounts for all functional requirements: each requirement has an implementation in a use case and a priority. UCs are shown with priority scores.

Requirements to Use Case Traceability Matrix

	Priority Weight	UC1	UC2	UC2.1	UC2.2	UC2.3	UC3	UC4	UC5
R1	1	X							
R2	3								X
R3	4		X						
R4	3			X					
R5	3			X					
R6	3				X				
R7	3				X	X			
R8	5					X			
R9	2						X		
R10	2							X	
	SCORE	1	4	6	6	8	2	2	3

NOTE: Priority 1 is highest priority, work this first

Increment Matrix

This is the team's plan to build the app. The priority is the same as the RUCTM, effort is estimated time to program the use case and 'depends on' relationships are noted. The iterations are timeframes in which the use cases are programmed, in priority order (i.e., most important use cases developed first, and so on). This iteration matrix is the team App Project Build Plan.

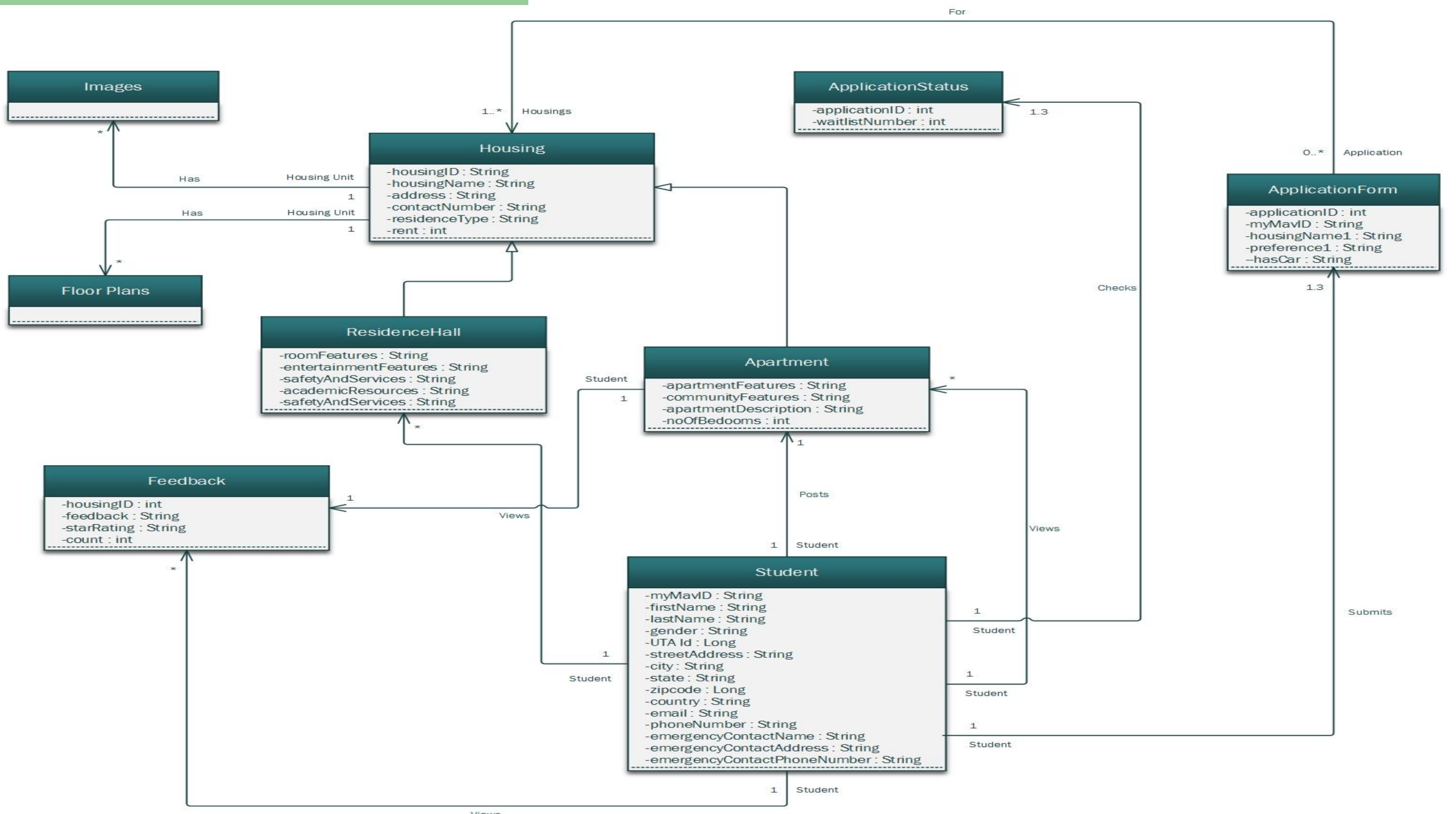
Use case	Priority	Effort *(person-weeks)*	Depends on	Assigned to	Iteration 1 *(Due Date)*	Iteration 2 *(Due Date)*	Iteration 3 *(Due Date)*
UC1	1	3	None	AB, PZ	3		
UC2	4	4	UC1	RR		3	1
UC2.1	6	2	UC1	TM		1	1
UC2.2	6	2	UC1, UC2, UC3	TM		1	1
UC2.3	8	2	UC1, UC2, UC3	AB		1	1
UC3	2	3	UC2	TM, RR	2	1	
UC4	2	1	UC3	PZ		1	
UC5	3	1	UC1	PZ		1	
Total Effort		18			5	9	4

1 Person-Week = 5 hrs.

Team Members: Ariel B., Tom M., Paul Z., Rene R.

Domain Model

Also known as a Context Diagram



Task List

- Organized list of UC development tasks and team assignments
- For each Increment

- Optional: UC programming assignments are contained in Increment Matrix “assigned to” column
  - May use task list to include and assign non-programming assignments

Iteration 2

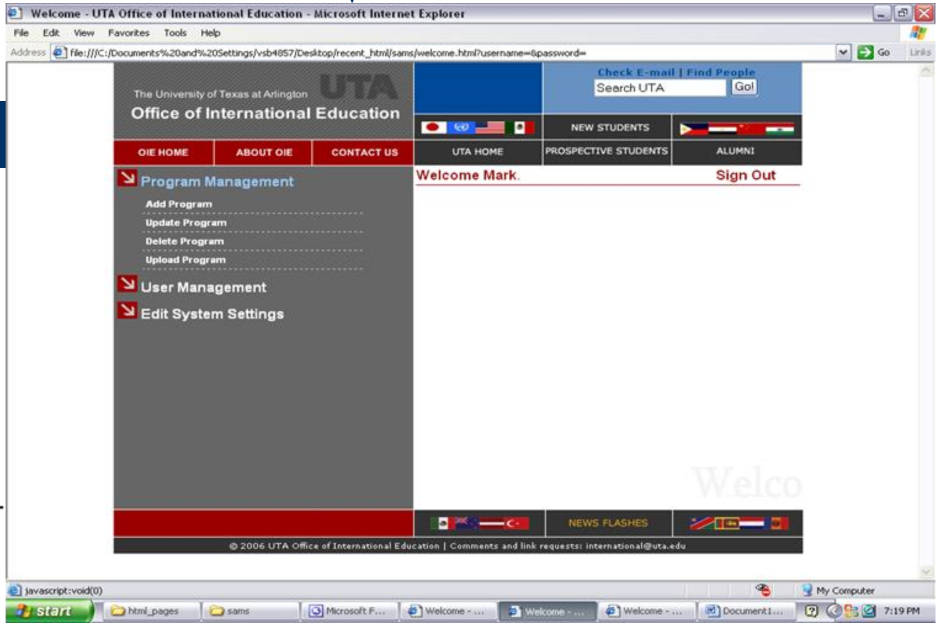
- Conduct design and analysis
- Add the following diagrams to previous iteration
  - Expanded Use Cases (per increment matrix planning)
    - Need an EUC for each UC
  - Design Sequence Diagrams for each EUC (with a non-trivial step)
  - Design Class Diagram

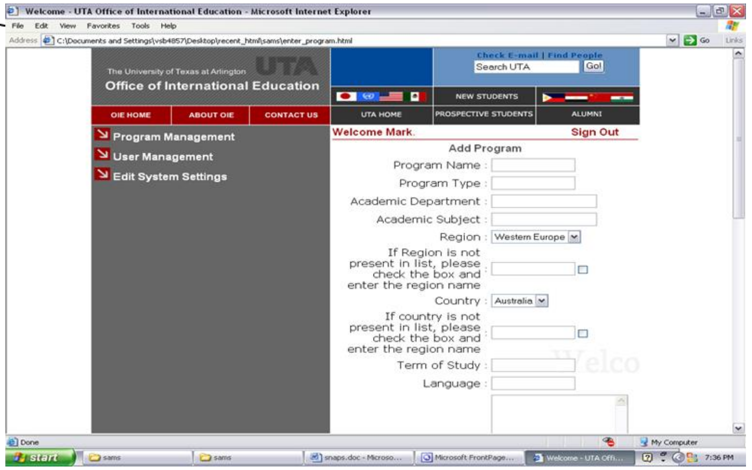
User Interface Prototypes (UIPs)
(Your UIPs will look like Phone screens)

Expanded Use Case Example

EUC2: Add Program

***Precondition**: This use case assumes that the staff user has logged into the system and is seeing the staff main page.*	
Actor: Staff User	System: SAMS
	0. System displays the staff main page.
1. **TUCBW** the staff user clicks the "Add Program" button.	2. System displays the Add Program page.
3. The staff user enters program detail and clicks the "submit" button.	4. System checks the submitted info and shows a confirmation message if no error is found.
5. **TUCEW** the staff user clicks the "OK" button on the confirmation page.	
***Postcondition**: The added program is immediately available for search.*	


Welcome - UTA Office of International Education - Microsoft Internet Explorer
File Edit View Favorites Tools Help
The University of Texas at Arlington
Office of International Education
Check E-mail | Find People
Search UTA Go!
OIE HOME ABOUT OIE CONTACT US UTA HOME PROSPECTIVE STUDENTS ALUMNI NEW STUDENTS
Program Management
Add Program
Update Program
Delete Program
Upload Program
User Management
Edit System Settings
Welcome Mark.
Sign Out
NEWS FLASHES
© 2006 UTA Office of International Education | Comments and link requests: international@uta.edu


Welcome - UTA Office of International Education - Microsoft Internet Explorer
Office of International Education
OIE HOME ABOUT OIE CONTACT US UTA HOME PROSPECTIVE STUDENTS ALUMNI
Program Management
User Management
Edit System Settings
Welcome Mark.
Sign Out

Add Program
Program Name :
Program Type :
Academic Department :
Academic Subject :
Region : Western Europe
If Region is not present in list, please check the box and enter the region name :
Country : Australia
If country is not present in list, please check the box and enter the region name :
Term of Study :
Language :

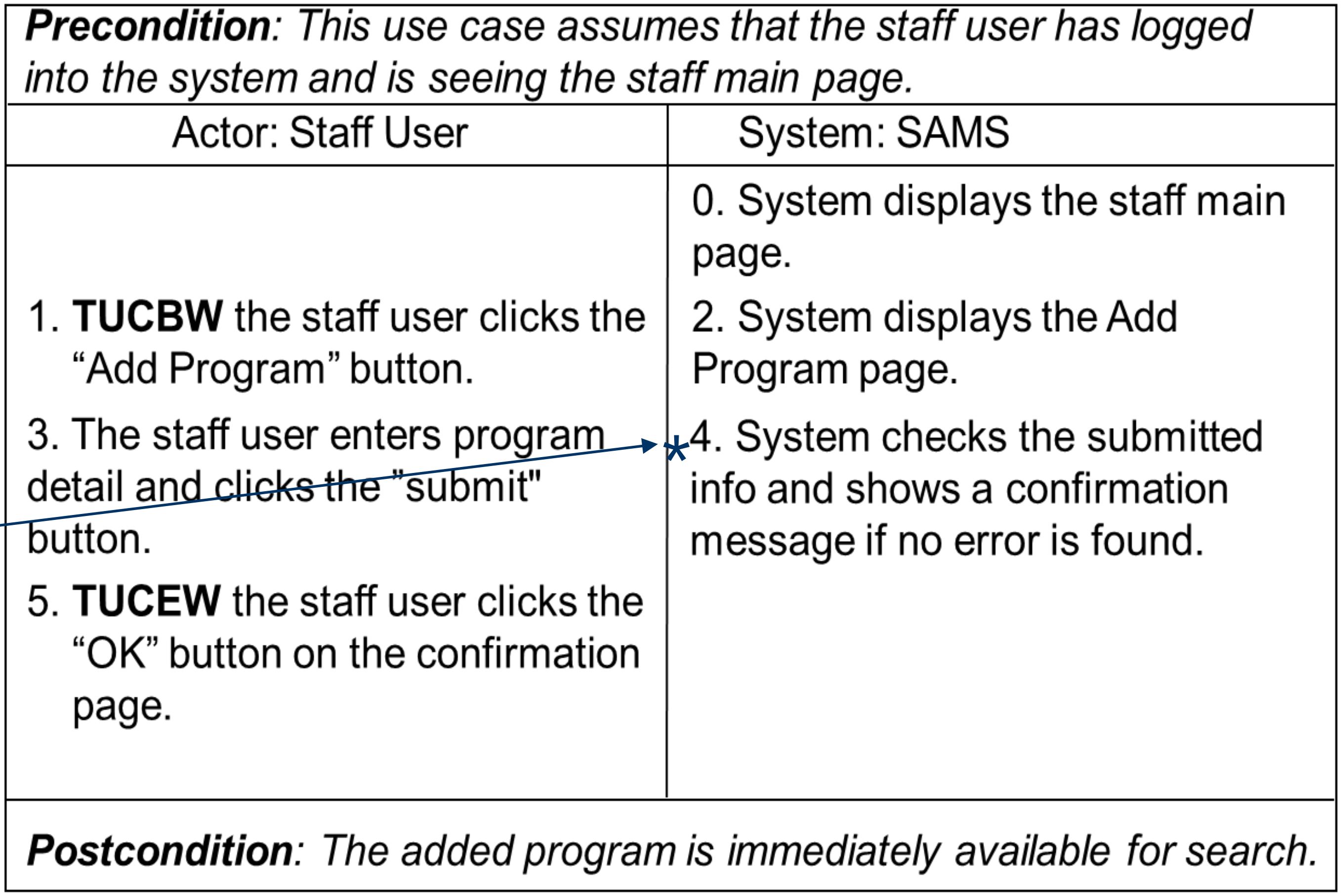
Identifying a Non-Trivial Step

- A trivial step is
  - If the step does not require background processing
  - If the system response simply displays a menu or input dialog
  - If the step displays the same system response for all actors

- A **nontrivial** step is
  - A system response that requires background processing
  - A system response that is different for different actors (i.e., not just a standard GUI response)
  - Key question: does the response require other objects to interact and collaborate with each other to fulfill the request? If yes, then this is a non-trivial step.

Non-Trivial Step

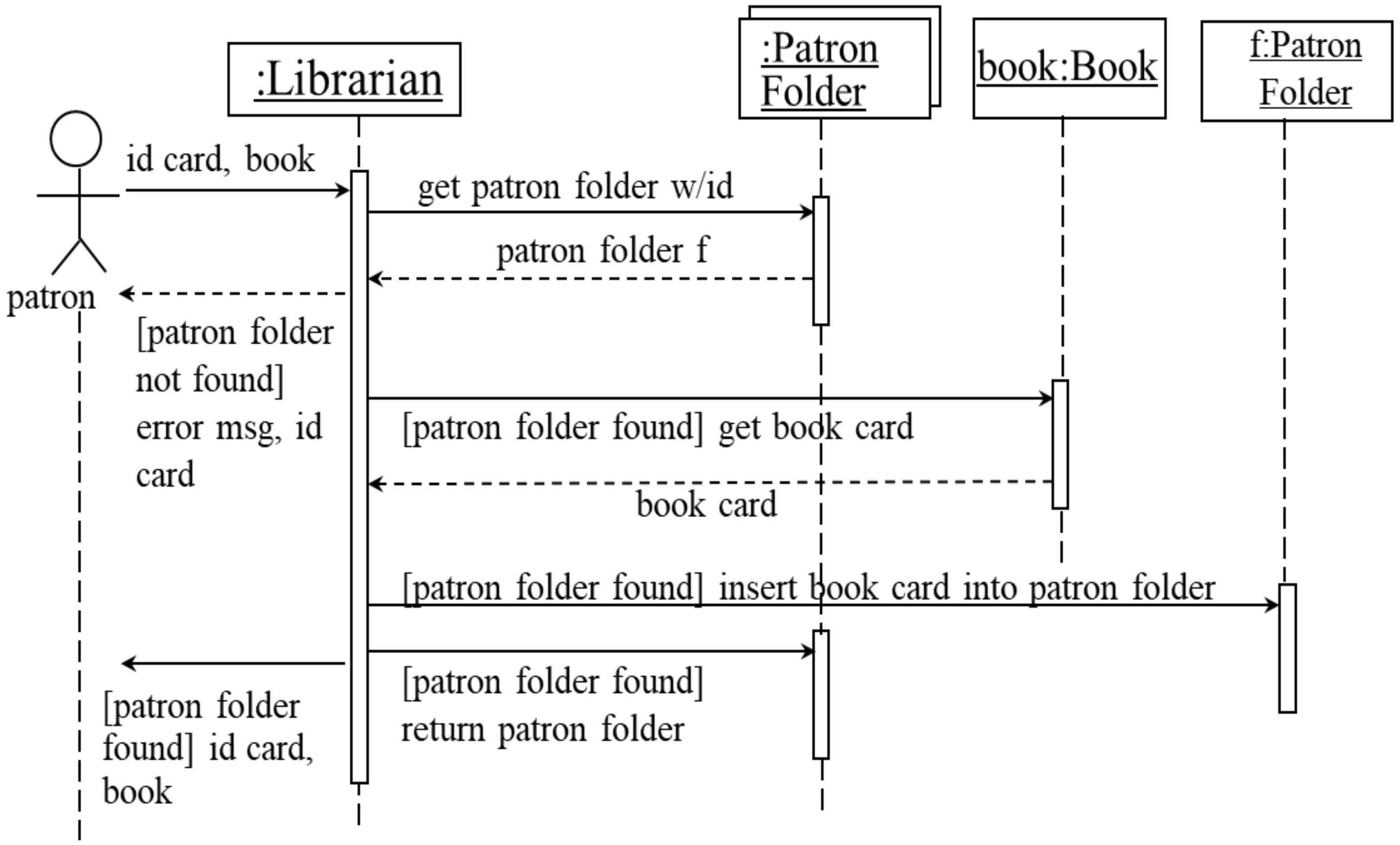
- Non-trivial Step
  - Does some background processing before fulfilling the request
  - Mark step with ‘ * ’



__Precondition__: This use case assumes that the staff user has logged into the system and is seeing the staff main page.	
Actor: Staff User	System: SAMS
	0. System displays the staff main page.
1. **TUCBW** the staff user clicks the “Add Program” button.	2. System displays the Add Program page.
3. The staff user enters program detail and clicks the "submit" button.	*4. System checks the submitted info and shows a confirmation message if no error is found.
5. **TUCEW** the staff user clicks the “OK” button on the confirmation page.	
__Postcondition__: The added program is immediately available for search.	

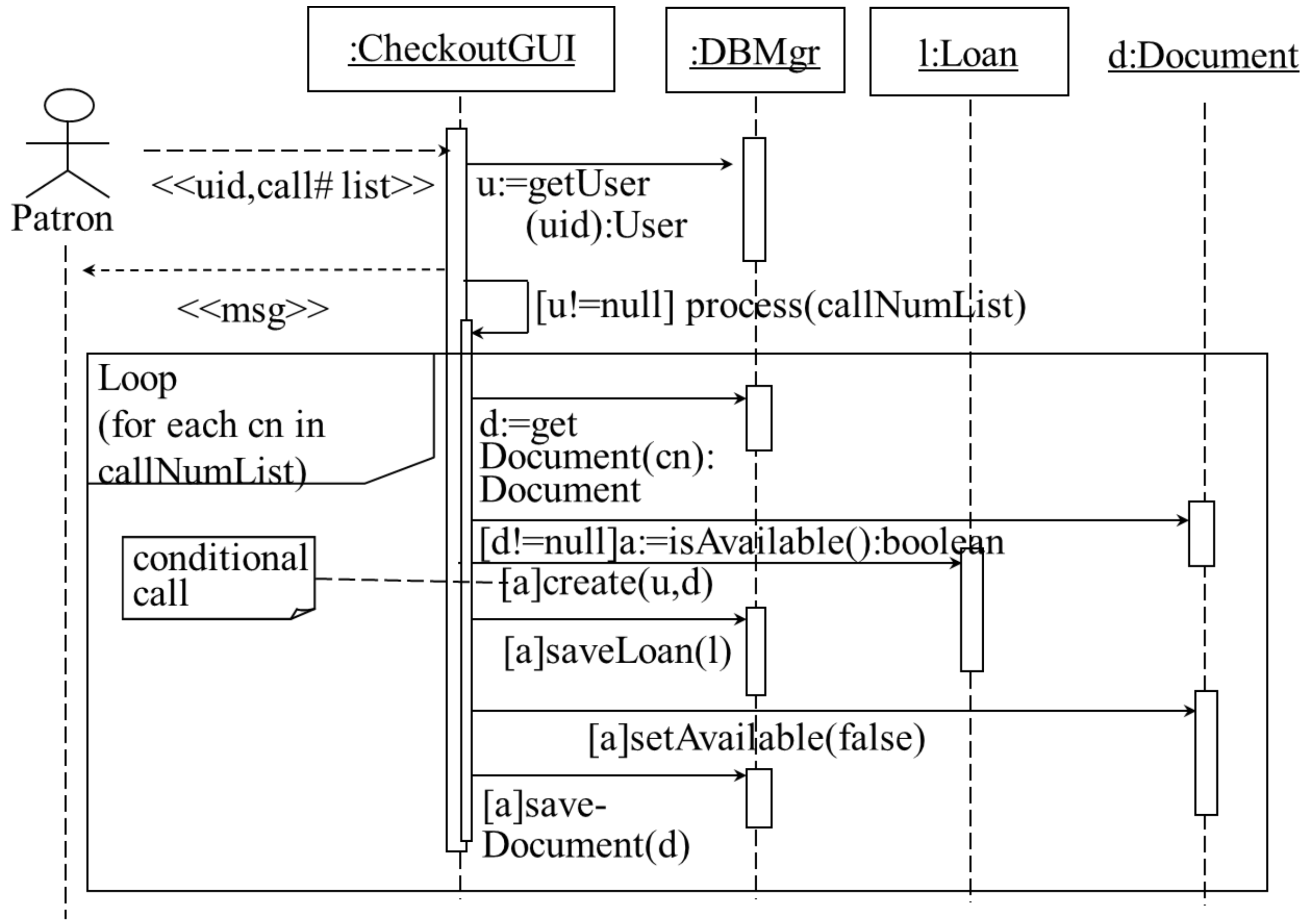
Analysis Sequence Diagram Example

- Analysis Sequence Diagram provides messages constructed in plain text for object interaction
- Good for initial design and 'what if' design investigations


:Librarian
:Patron Folder
book:Book
f:Patron Folder
patron
id card, book
get patron folder w/id
patron folder f
[patron folder not found] error msg, id card
[patron folder found] get book card
book card
[patron folder found] insert book card into patron folder
[patron folder found] return patron folder
[patron folder found] id card, book

Design Sequence Diagram

- Design SD includes details of messages
  - includes ‘code’ associated with implementing methods
- A sequence diagram is needed for each higher-risk use case (i.e., EUCs with non-trivial steps)
- For your project, create design sequence diagrams



Design Class Diagram

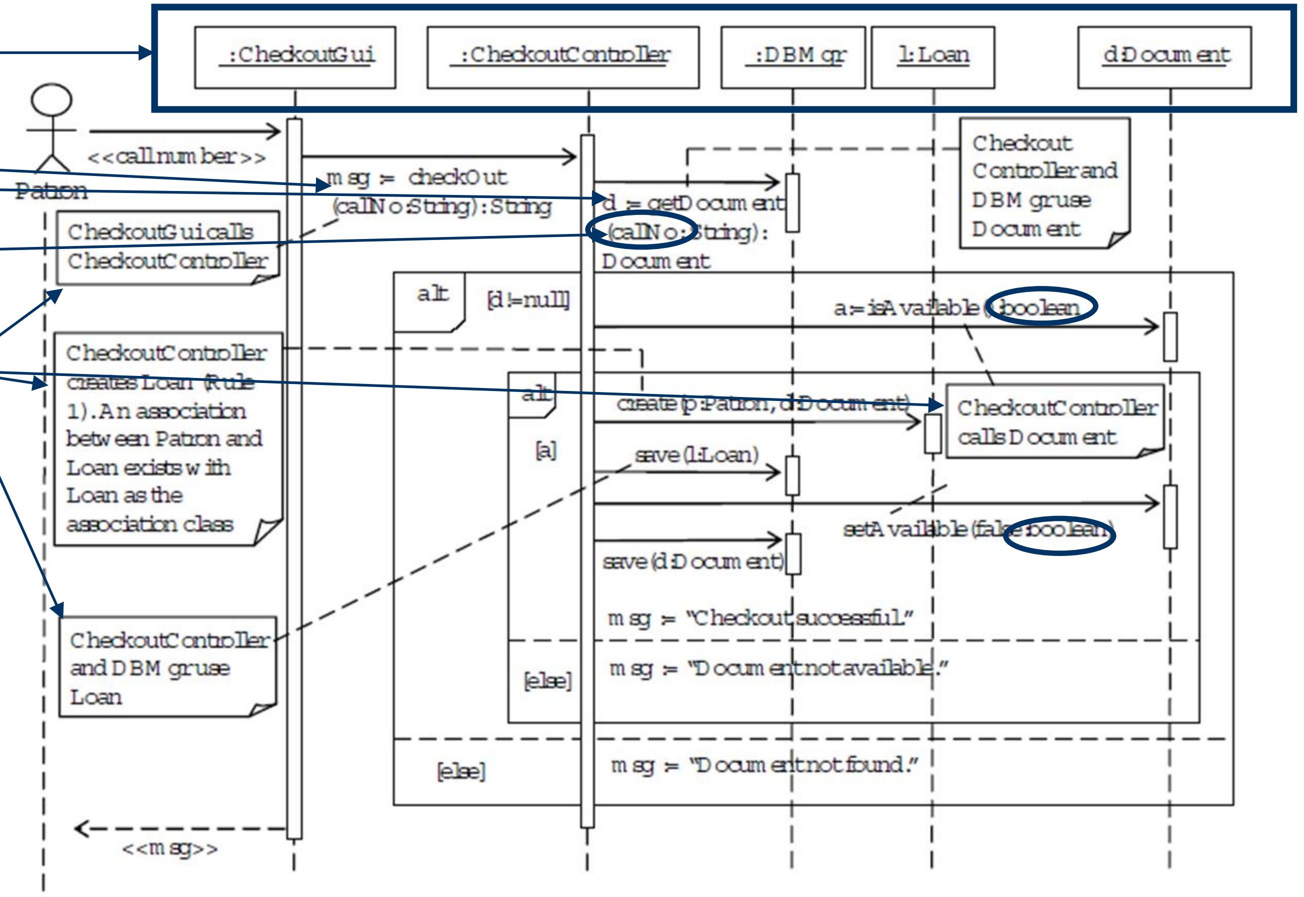
- A Design Class Diagram (DCD) is a structural diagram.
- It shows the classes, their attributes and operations, and relationships between the classes. It may also show the design patterns used.
- It is used as a basis for implementation, testing, and maintenance.
- The collection of the domain model and sequence diagrams does not provide a class structure and road-map to guide subsequent efforts. Information is contained in these diagrams, but is scattered across them.
- The diagram that provides this design specification of the classes and the class structure is the DCD.
- There is only one DCD for the system.

Steps to Create Domain Class Diagram

1. Identify Classes
   - from the Design Sequence Diagrams for the current iteration
2. Identify Methods
   - from the Design Sequence Diagrams for the current iteration
3. Identify Attributes
   - From the Design Sequence Diagrams and earlier Domain Model
4. Assess relationships between classes and their implications
   - Call, use, or create relationships

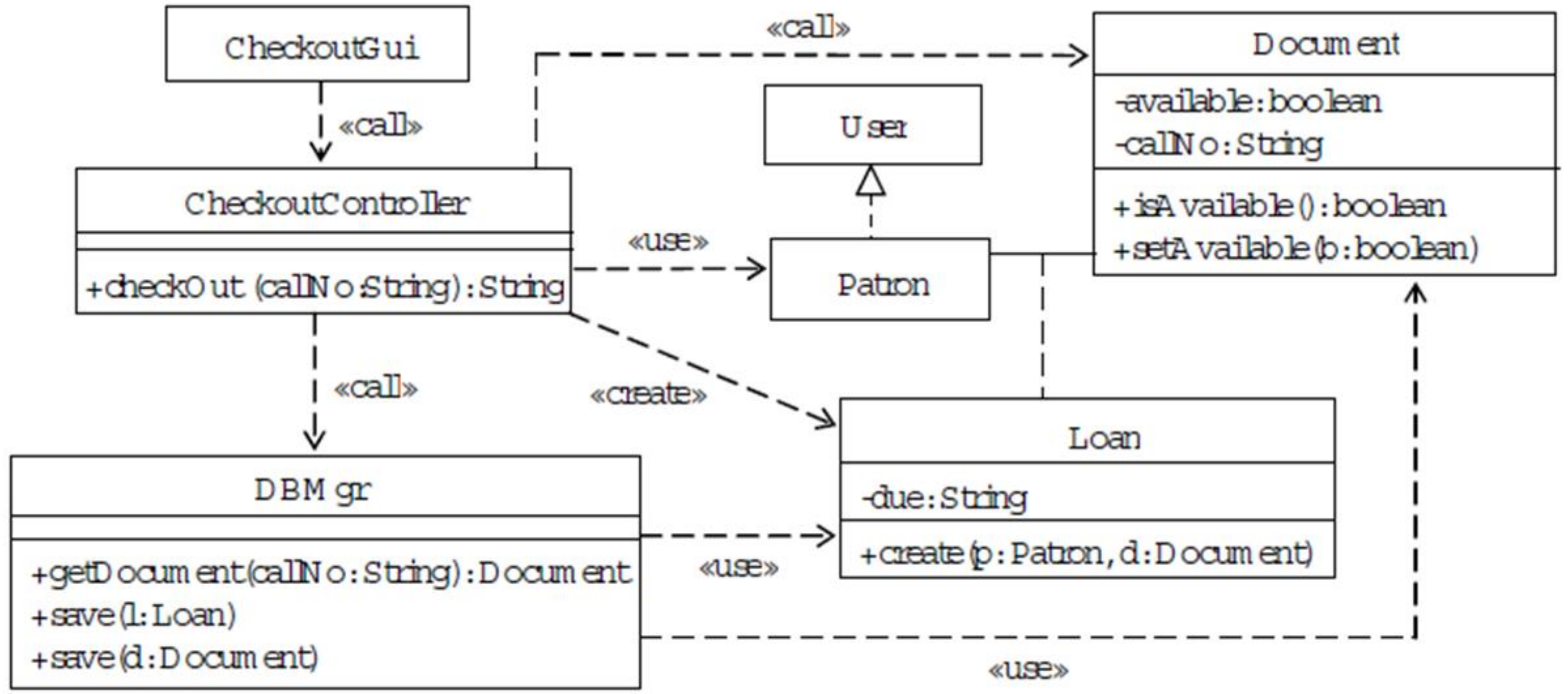
Example Design Sequence Diagram to DCD

- Identify Classes
- Identify Methods
- Identify Attributes
- Identify Relationships



:CheckoutGui
:CheckoutController
:DBMgr
l:Loan
d:Document
Patron
<<callnumber>>
msg := checkOut(callNo:String):String
d := getDocument(callNo:String): Document
Checkout Controller and DBMgr use Document
CheckoutGui calls CheckoutController
alt
[d!=null]
a:=isAvailable():boolean
CheckoutController creates Loan (Rule 1). An association between Patron and Loan exists with Loan as the association class
alt
create(p:Patron, d:Document)
CheckoutController calls Document
[a]
save(l:Loan)
setAvailable(false):boolean
save(d:Document)
msg := "Checkout successful."
[else]
msg := "Document not available."
CheckoutController and DBMgr use Loan
[else]
msg := "Document not found."
<<msg>>

Example DCD


CheckoutGui
«call»
CheckoutController
+checkOut (callNo:String):String
«call»
Document
-available:boolean
-callNo:String
+isAvailable():boolean
+setAvailable(b:boolean)
User
«use»
Patron
«create»
«call»
Loan
-due:String
+create(p:Patron,d:Document)
DBMgr
+getDocument(callNo:String):Document
+save(l:Loan)
+save(d:Document)
«use»
«use»

Iteration 3

- Conduct analysis and Implementation
  - Additional Design Models as needed
    - Activity diagrams, state charts, etc.
  - Update any drawings needed
- Create you-tube demonstration of app showing all use cases
  - Provide you-tube URL in final iteration submittal package

App Demo

- Create a you-tube video of your app demo and turn in URL with Iteration 3 report submission
  - Should demonstrate each UC, one at a time in succession
- Demo app in class using Android Phone emulator

  -- OR --
- Demo app using Android App downloaded to phone

- Don’t wait until last minute AND don’t keep tweaking your app!
  - Good enough is probably good enough!