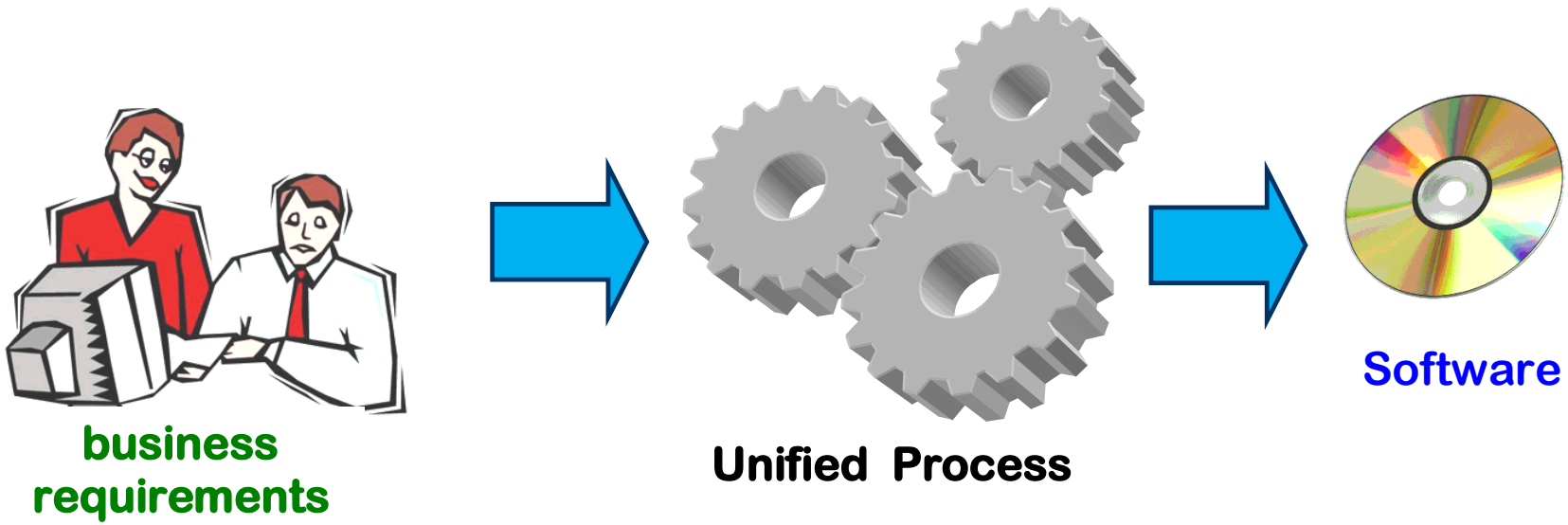
UNIVERSITY OF PISA
COMPUTER ENGINEERING – SOFTWARE SYSTEMS ENGINEERING

UML and Unified Process

Mario GCA Cimino

2. Unified Process

✓ A Software Development Process (**SDP**) defines the *who*, *what*, *when*, and *how* of developing software. The Unified Process (**UP**) is an industry standard SDP from the authors of the **UML** (Unified Modeling Language).



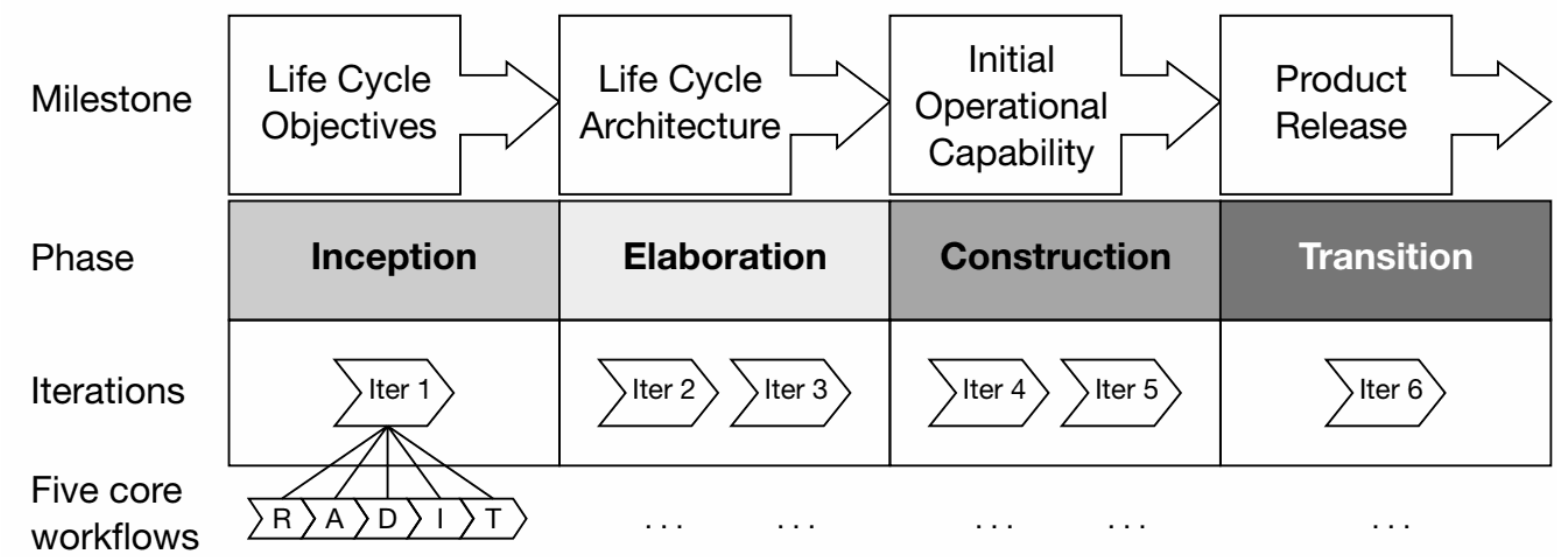
✓ UP is ***iterative*** and ***incremental***: a large software development project is broken down into smaller "mini projects" called ***iterations***. Each iteration generates a more complete version of the final system. The difference between two consecutive versions is called ***increment***.

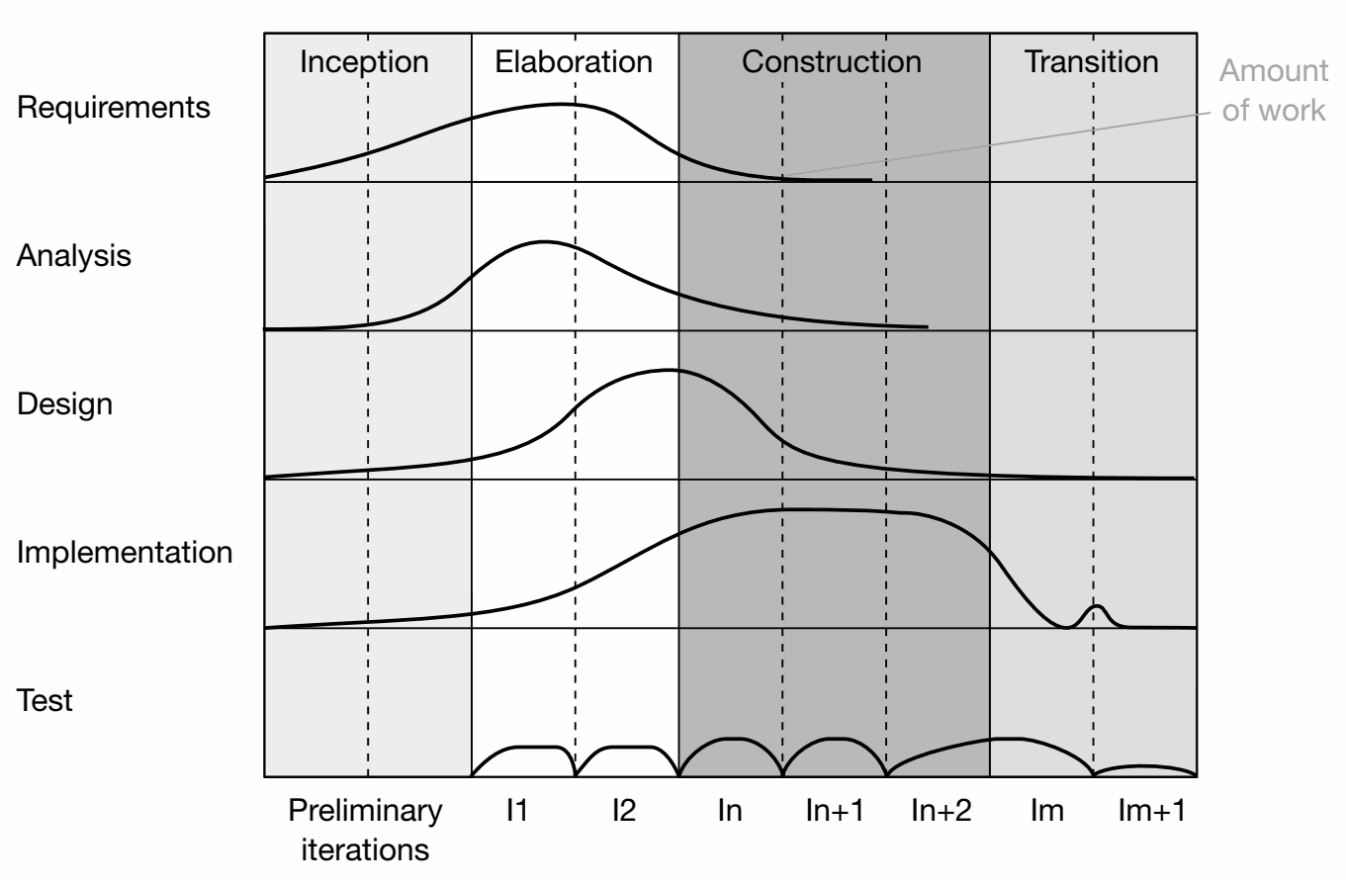
✓ Each iteration is made by five *core workflows*, with different emphasis:

- **R: requirements** – capturing what the system should do;
- **A: analysis** – refining and structuring the requirements;
- **D: design** – realizing the requirements in system architecture;
- **I: implementation** – building the software;
- **T: test** – verifying that the implementation works as desired.

✓ In a team work, it is often convenient to schedule iterations in parallel, according to dependencies between the artifacts of each iteration.

✓ UP consists of a sequence of four *phases*, terminating with related milestones:





UP: core workflows versus phases

- ✓ **Inception**: most of the work in early requirements and analysis
  Elaboration: the emphasis on requirements and analysis and some design
  Construction: mostly design and implementation, with related testing
  Transition: residual implementation and test

PHASE	GOALS	FOCUS	MILESTONE
Inception	● *capturing essential requirements* to help scope the system ● feasibility: technical prototype to validate technology, proof of concept to validate business requirements ● business case to demonstrate that the project will deliver quantifiable business benefit	● *requirements and analysis workflows* ● some design and implementation, to build technical prototype or proof of concept ● no testing – throwaway prototype	● Life Cycle Objectives (requirements/ features/constraints, initial use cases) → see conditions and deliverable table
Elaboration	● *create executable architectural baseline* ● *capture use cases* to 80% functional requirements; ● refine the Risk Assessment; ● define quality attributes (defect discovery rates, acceptable defect densities, etc.); ● create detailed plan for construction; ● formulate a bid that includes resources, time, equipment, staff and cost.	● *requirements, analysis and design workflows* ● implementation: build the initial operational capability ● test the initial operational capability (alpha test, internal)	● Life Cycle Architecture
Construction	● *complete requirements, analysis and design* ● move from architectural baseline to the *final system* ● maintain the system architecture integrity	● *implementation and testing* ● build the Initial Operational capability ● test the Initial Operational Capability	● Initial Operational Capability (software system is finished for beta testing in productive environment)
Transition	● starts after beta testing is completed and the system is finally deployed ● correct defects, prepare the user site for the new software; ● create user manuals and other documentation; provide user consultancy; ● conduct a post project review	● *no requirements, analysis* ● *finish implementation and complete test workflows* ●modify design if problems arise in beta testing ● user acceptance testing (user community)	● Product Release (the product is accepted into the user community)

Inception: conditions to attain for the Life Cycle Objectives

Conditions of satisfaction	Deliverable
The stakeholders have agreed the project objectives	A vision document that states the project's main requirements, features and constraints
System scope has been defined and agreed with the stakeholders	An initial use case model (only about 10% to 20% complete)
Key requirements have been captured and agreed with the stakeholders	A Project Glossary
Cost and schedule estimates have been agreed with the stakeholders	An initial Project Plan
A business case has been raised by the project manager	Business Case
The project manager has performed a risk assessment	A Risk Assessment document or database
Confirmation of feasibility through technical studies and/or prototyping	One or more throwaway prototypes
An outline architecture	An initial architecture document

Elaboration: conditions to attain for the Life Cycle Architecture

Conditions of satisfaction	Deliverable
A resilient, robust executable architectural baseline has been created	The executable architectural baseline
The executable architectural baseline demonstrates that important risks have been identified and resolved	UML Static Model UML Dynamic Model UML Use Case Model
The vision of the product has stabilized	Vision document
The risk assessment has been revised	Updated Risk Assessment
The business case has been revised and agreed with the stakeholders	Updated Business Case
A project plan has been created in sufficient detail to enable a realistic bid to be formulated for time, money and resources in the next phases	Updated Project Plan
The stakeholders agree to the project plan	
The business case has been verified against the project plan	Business Case and Project Plan
Agreement is reached with the stakeholders to continue the project	Sign-off document

Construction: conditions to attain for the Initial Operational Capability

Conditions of satisfaction	Deliverable
The software product is sufficiently stable and of sufficient quality to be deployed in the user community	The software product The UML model Test suite
The stakeholders have agreed and are ready for the transition of the software to their environment	User manuals Description of this release
The actual expenditures vs. the planned expenditures are acceptable	Project Plan

Transition: conditions to attain for the Product Release

Conditions of satisfaction	Deliverable
Beta testing is completed, necessary changes have been made, and the users agree that the system has been successfully deployed	The software product
The user community is actively using the product	
Product support strategies have been agreed with the users and implemented	User support plan User manuals

Power Driver: vision document

Power Driver is a vehicle rental company. Power Driver has 105 branches over the world. Currently there is no linkage between these branches. Power Driver needs a computer system to connect all these branches together to support better services to the customer in a more efficient way. Following are the requirements of the computer system.

The system should be able to store the vehicle records. A vehicle record includes model number, serial number, status, booking schedule and booking history. The status of the vehicle should be available, rented, repairing or reserved. Staffs only allow in searching the vehicle records. Only the branch's manager allows in maintaining vehicle records. The system stores the customer records. The customer record includes name, ID, status and rental history. The status of the customer can be normal, VIP or suspend. The normal and VIP customers can rent vehicles but suspended customers are not allowed to rent vehicles. All staffs can maintain customer records. But only the branch's manager allows changing the status of the customers. Each branch has one manager.

The system should provide a function for staffs to input the rental record. The staff can only create the rental record for the car in his own working branch. The staff cannot create a rental record for other branches. After the vehicle is returned, the staff needs to update the rental record. The system should have a reserved car function for booking the car. The start of the booking period should be within

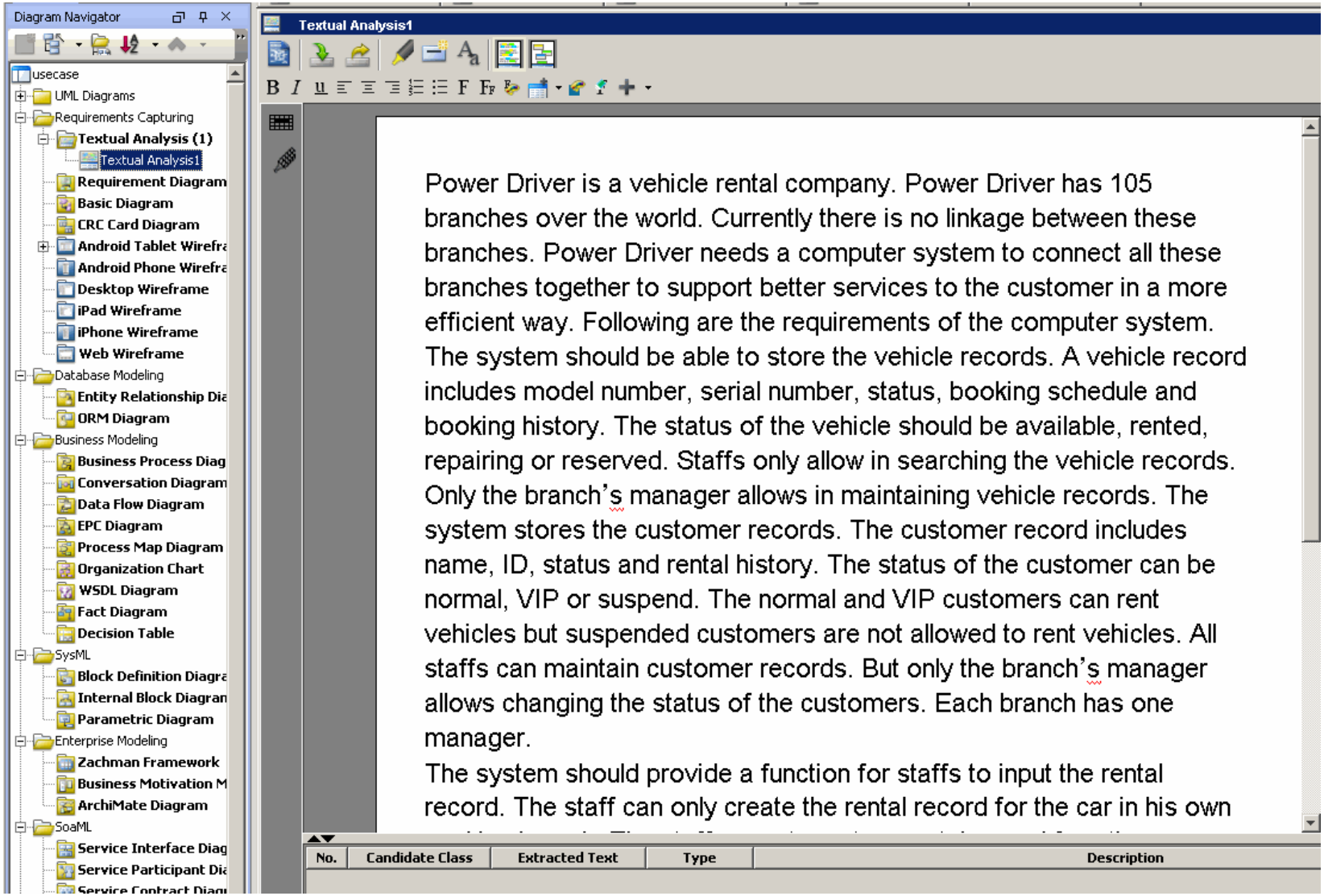
next 7 days. Normal staff cannot make a booking request to another branch. Only the manager can request a booking to another branch.

The staff can use the system to check the availability of the car by using model number, or booking schedule. If there are suitable car(s) for renting, the staff can use the system to reserve or rent the car. If the car is rented, other staffs should not rent the car to another customer until the rental period is over. The staff can search the customer records by using customer's name, ID or telephone.

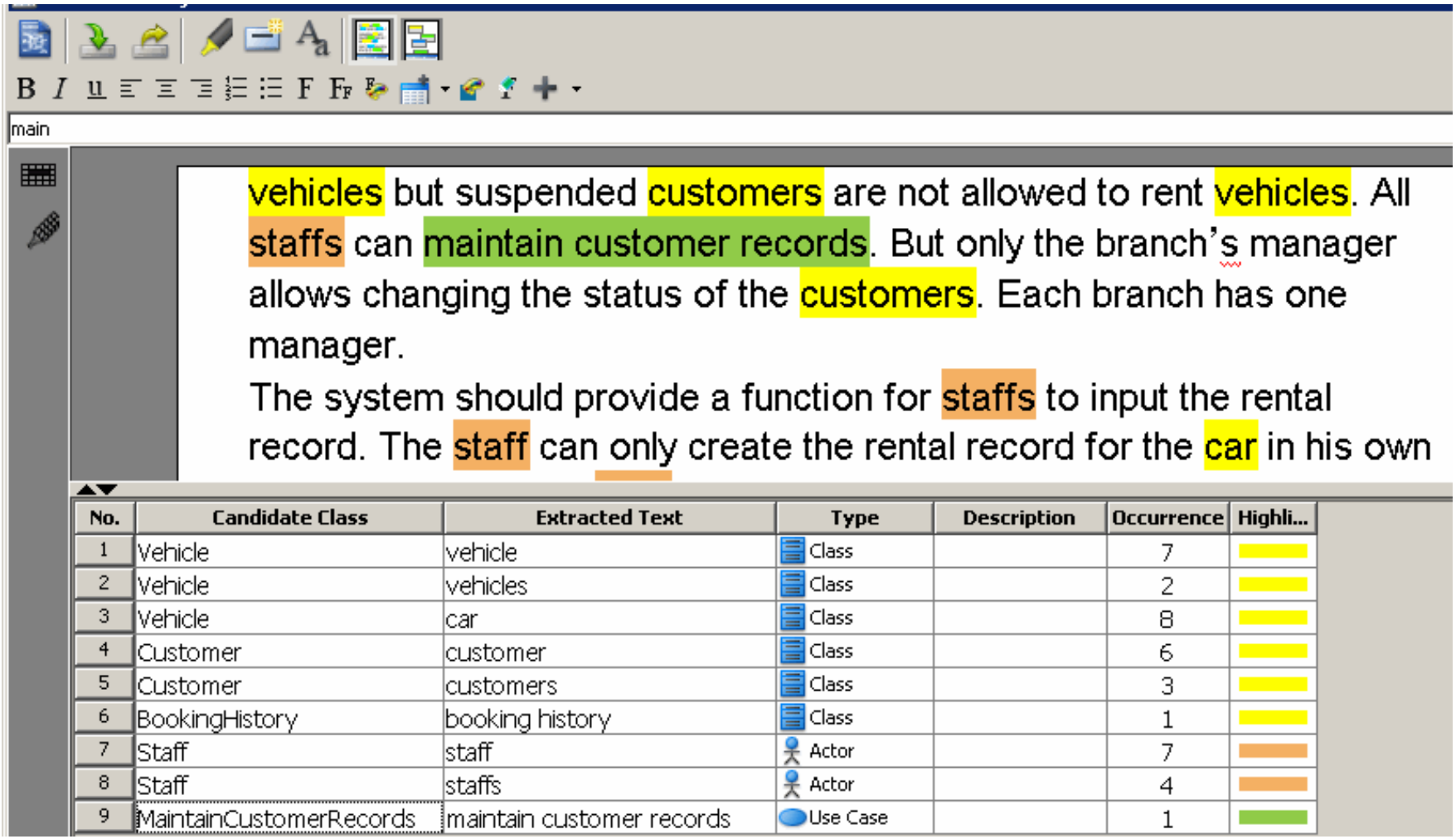
The system should provide a report generation function to generate a monthly report of the branch.

1. Textual Analysis and candidate items

1. Diagram Navigator \ Requirements Capturing \
2. Right click on Textual Analysis → *New Textual Analysis*
3. Paste or import textual description



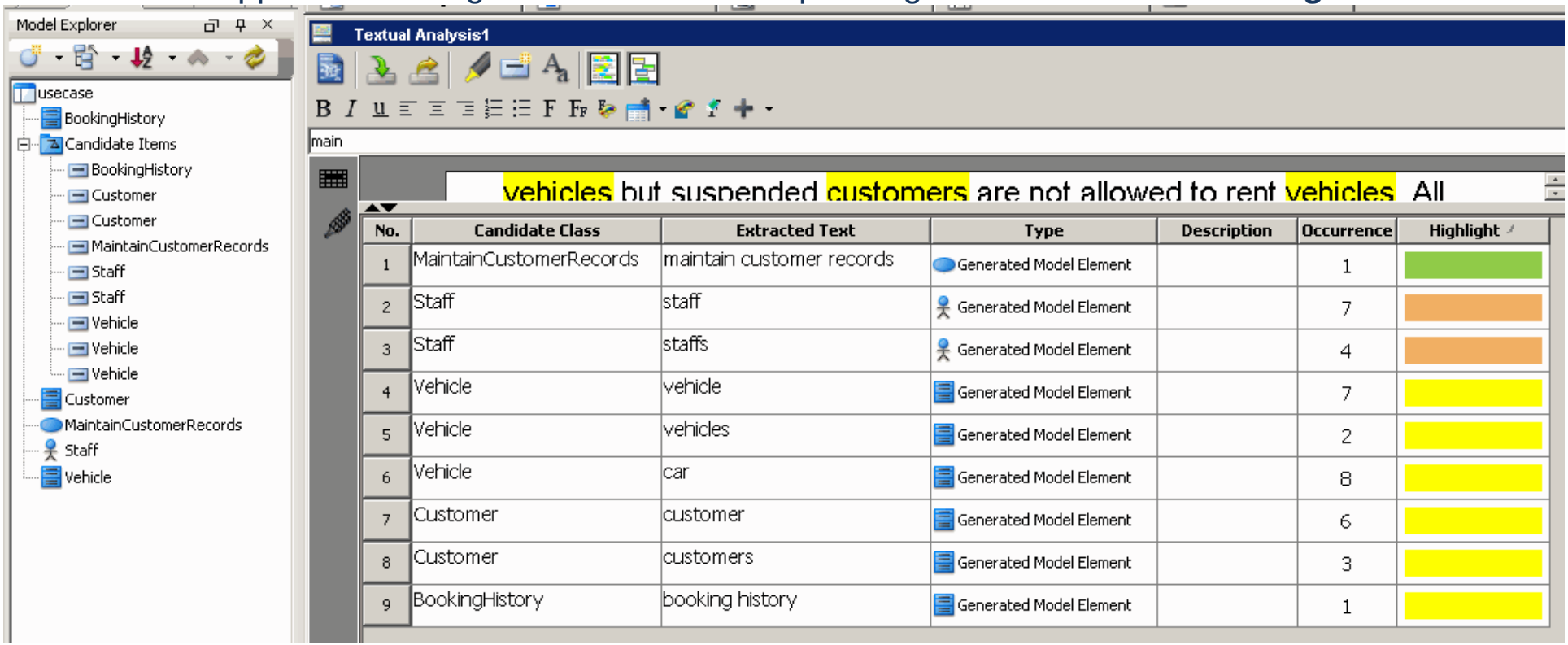
4. Right click on the term *vehicle, vehicles, car, cars* → Class
5. Right click on the term *customer , customers* → Class
6. Right click on the text *booking history* → Class
7. Right click on the term *staff, staffs* → Actor *Staff*
8. In the underlying table, make uniform the field *candidate class* for singular/plural forms and synonyms by using the *camel case format* (e.g. *booking history* → *BookingHistory*)
9. Right click on the text *maintain customer records* → Use case *MaintainCustomerRecords*



No.	Candidate Class	Extracted Text	Type	Description	Occurrence	Highli...
1	Vehicle	vehicle	Class		7	
2	Vehicle	vehicles	Class		2	
3	Vehicle	car	Class		8	
4	Customer	customer	Class		6	
5	Customer	customers	Class		3	
6	BookingHistory	booking history	Class		1	
7	Staff	staff	Actor		7	
8	Staff	staffs	Actor		4	
9	MaintainCustomerRecords	maintain customer records	Use Case		1	

2. Model Elements

1. Right click on the row number corresponding to a candidate element → *Create Class Model Element → Do not visualize → close.*
2. If a model element with the same name already exists, the system asks whether the existing model element can be used. For synonym, plural/singular forms, select *yes*
3. Generated elements appear in the *Model Explorer* tab (on the left).
4. A model element which does not appear in any diagram is called ***hidden*** element. A classifier which appears in a diagram without a corresponding model element is called ***ghost*** *element*.

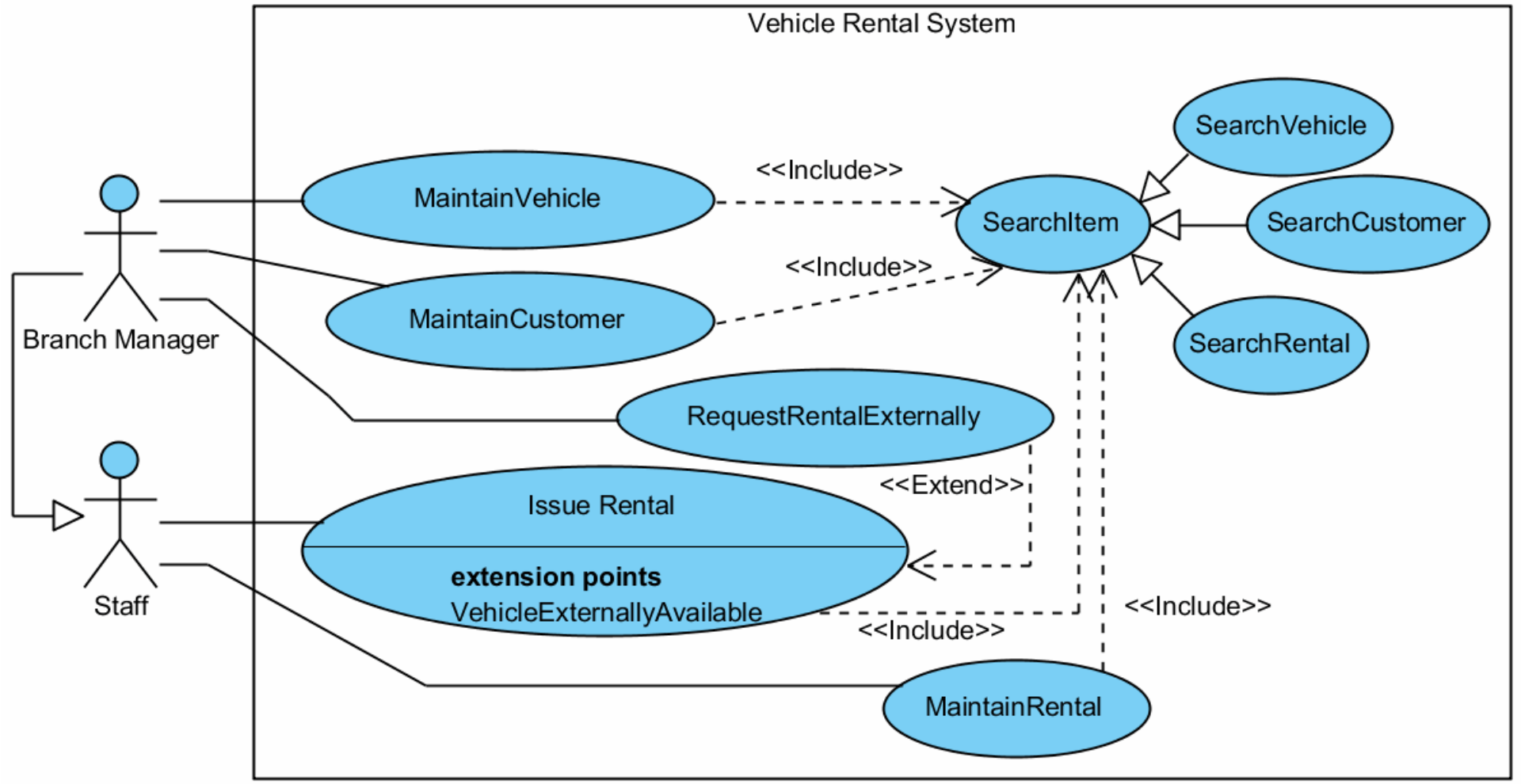


No.	Candidate Class	Extracted Text	Type	Description	Occurrence	Highlight
1	MaintainCustomerRecords	maintain customer records	Generated Model Element		1	
2	Staff	staff	Generated Model Element		7	
3	Staff	staffs	Generated Model Element		4	
4	Vehicle	vehicle	Generated Model Element		7	
5	Vehicle	vehicles	Generated Model Element		2	
6	Vehicle	car	Generated Model Element		8	
7	Customer	customer	Generated Model Element		6	
8	Customer	customers	Generated Model Element		3	
9	BookingHistory	booking history	Generated Model Element		1	

5. Model elements can be dragged to diagrams, generating views.

3. Use Case Diagram

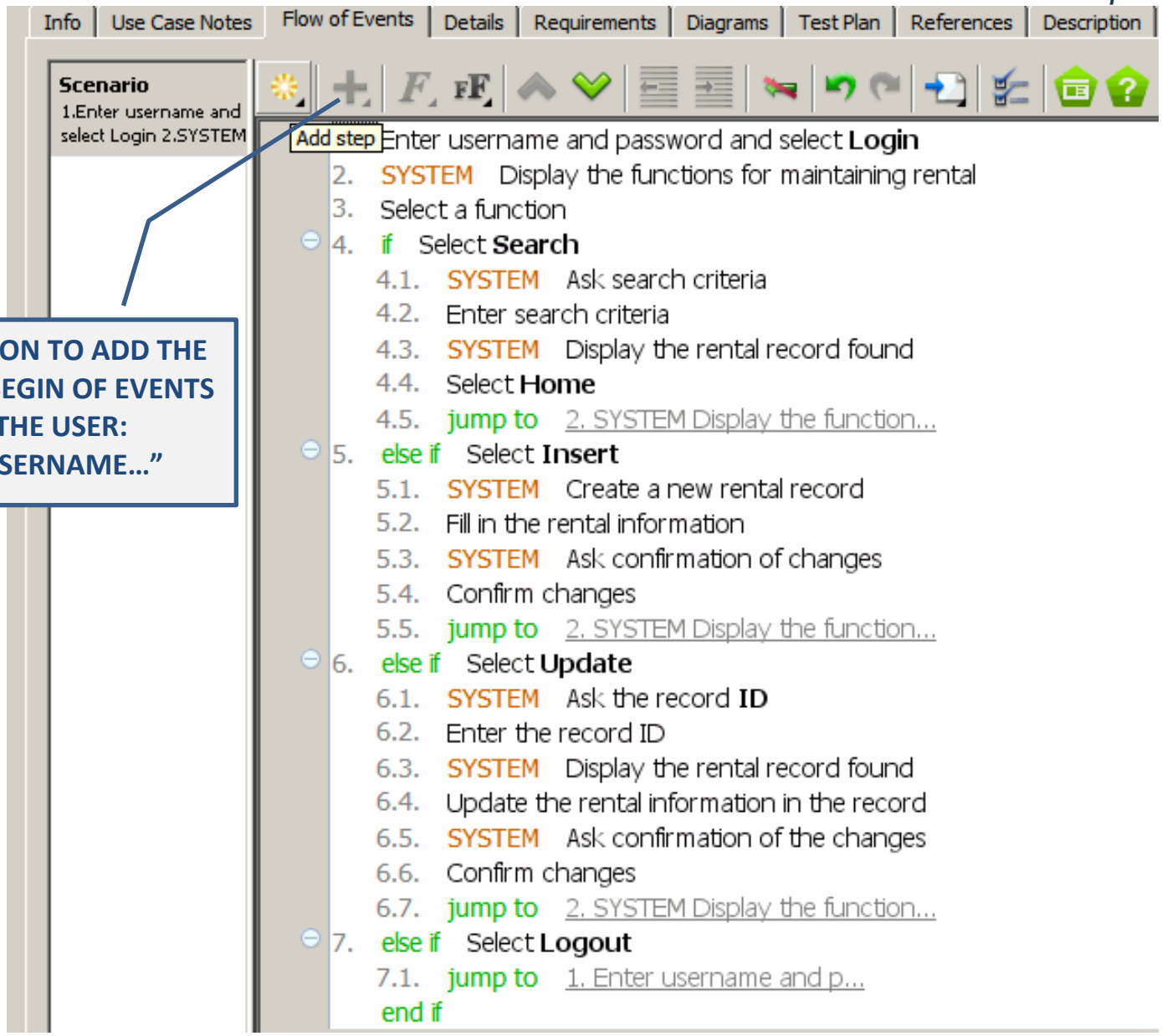
1. Right click on the Diagram Navigator \ Use case Diagram → *New Use Case Diagram.*
2. Drag the actor and the use case model elements *Staff* and *MaintainCustomerRecords* from the Model Explorer to the Use Case Diagram
3. Rename *MaintainCustomerRecords* to *MaintainCustomer*
4. Create new elements in the diagram using the toolkit (on the left).
5. To create a relationship between elements: hover mouse over an element, choose the relationship, drag the chosen relationship to the other element.
6. Define *BranchManager* as an extension of *Staff* (a *manager* is as an employee with other additional capabilities. A manager can take the place of any employee (*substitutability principle* in object-oriented models). As an effect, ***redundancy*** in the diagram is reduced.
7. Define an extension point over the *IssueRental* use case, entitled *VehicleExternallyAvailable* connected to *RequestRentalExternally*
8. Add a *SearchItem* use case, included by the other use cases, with three specializations *SearchVehicle, SearchCustomer* and *SearchRental*.



J. Arlow, *UML and the Unified Process*, pages to read:

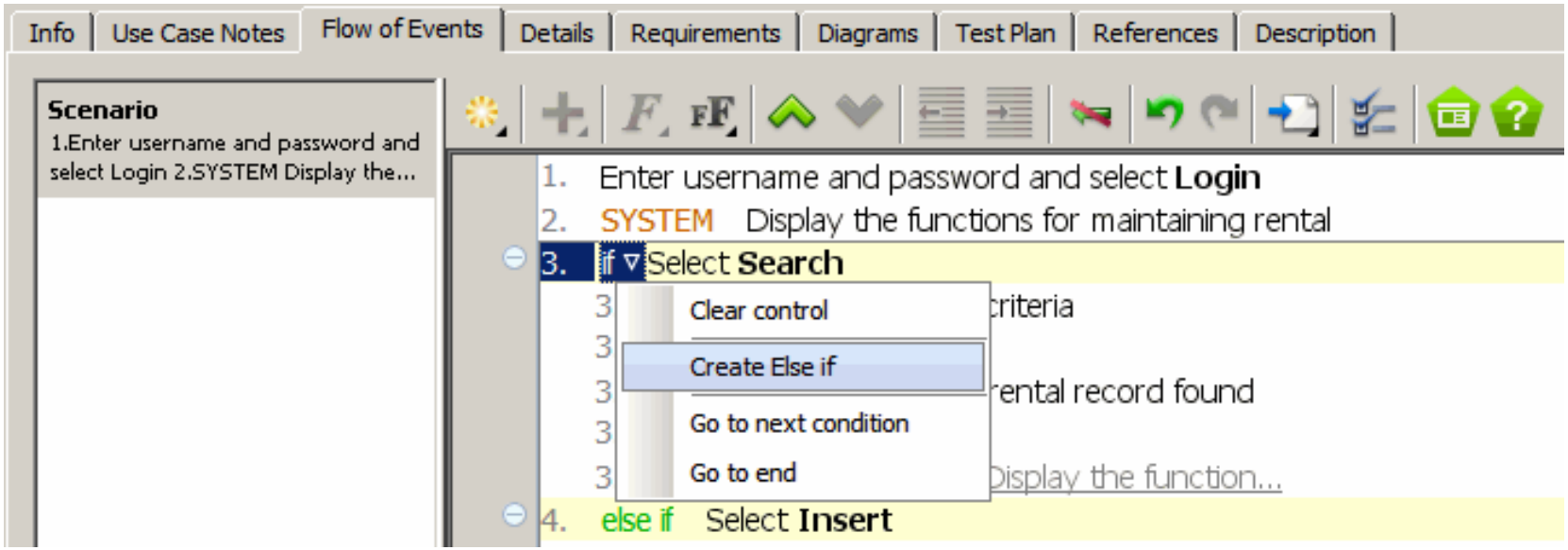
- from page 58 (Section 4.3.2) to page 62 (Section 4.3.3)
- from page 79 (Section 5.2) to page 82 (Section 5.3)
- from page 84 (Section 5.4) to page 88 (Section 5.5.1)

1. Right click on the MaintainRental use case → *Open UseCase Details…*
2. Tab *Flow of events* → Insert the main flow. For alternative flows use the *add step* button:

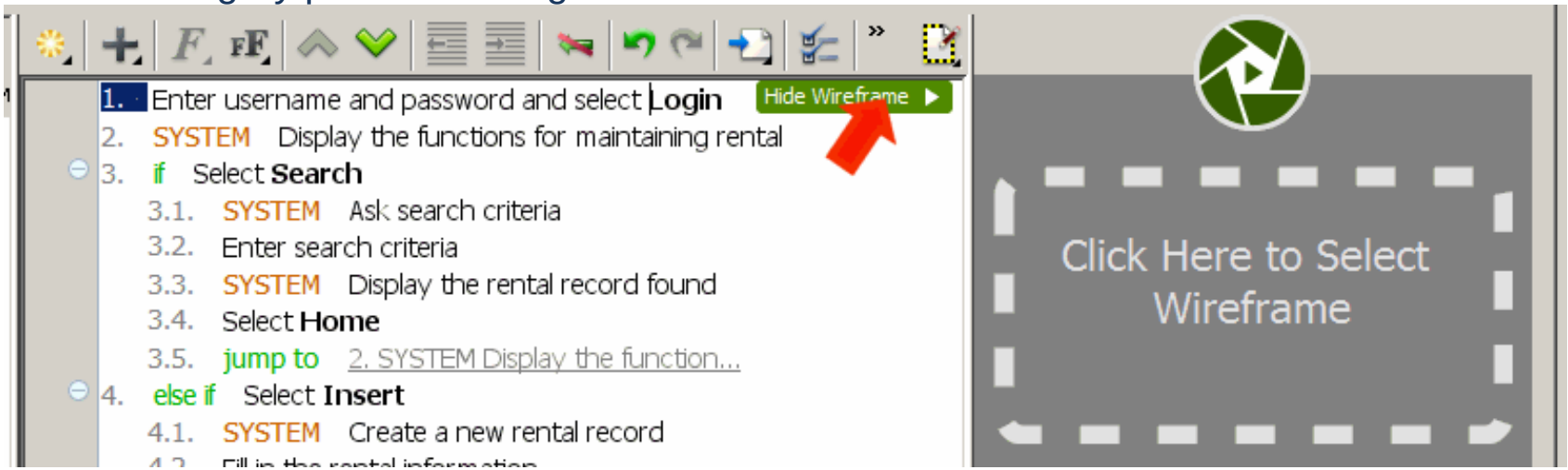


CLICK THE "+" ICON TO ADD THE ACTOR AT THE BEGIN OF EVENTS GENERATED BY THE USER: "STAFF ENTER USERNAME…"

3. To *create* an *Else* branch, hover mouse over an *If* branch and select *Create Else*.



4. To Create a Wireframe (screen flow of “mockups”, or “sketches” of the interface) move the mouse pointer over the green Wireframe button on the right hand side of the step. Click on it.
5. This shows a gray pane on the right hand side. Click on it to select a kind of wireframe to create



Flow of Events | Details | Requirements | Diagrams | Test Plan | References | Description

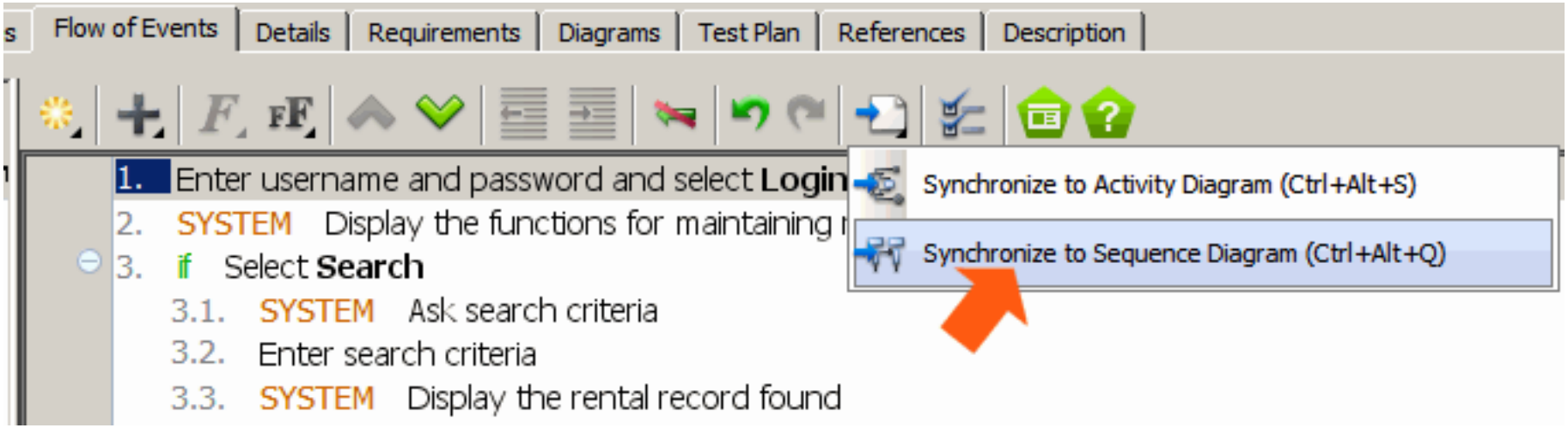
1. Enter username and password and select **Login**
2. SYSTEM Display the functions for maintaining ren
3. Select a function
4. if Select **Search**
   - 4.1. SYSTEM Ask search criteria
   - 4.2. Enter search criteria
   - 4.3. SYSTEM Display the rental record found
   - 4.4. Select **Home**
   - 4.5. jump to 2. SYSTEM Display the function...
5. else if Select **Insert**
   - 5.1. SYSTEM Create a new rental record
   - 5.2. Fill in the rental information
   - 5.3. SYSTEM Ask confirmation of changes
   - 5.4. Confirm changes
   - 5.5. jump to 2. SYSTEM Display the function...
6. else if Select **Update**
   - 6.1. SYSTEM Ask the record **ID**
   - 6.2. Enter the record ID
   - 6.3. SYSTEM Display the rental record found
   - 6.4. Update the rental information in the record
   - 6.5. SYSTEM Ask confirmation of the changes
   - 6.6. Confirm changes
   - 6.7. jump to 2. SYSTEM Display the function...
7. else if Select **Logout**
   - 7.1. jump to 1. Enter username and p...

   end if

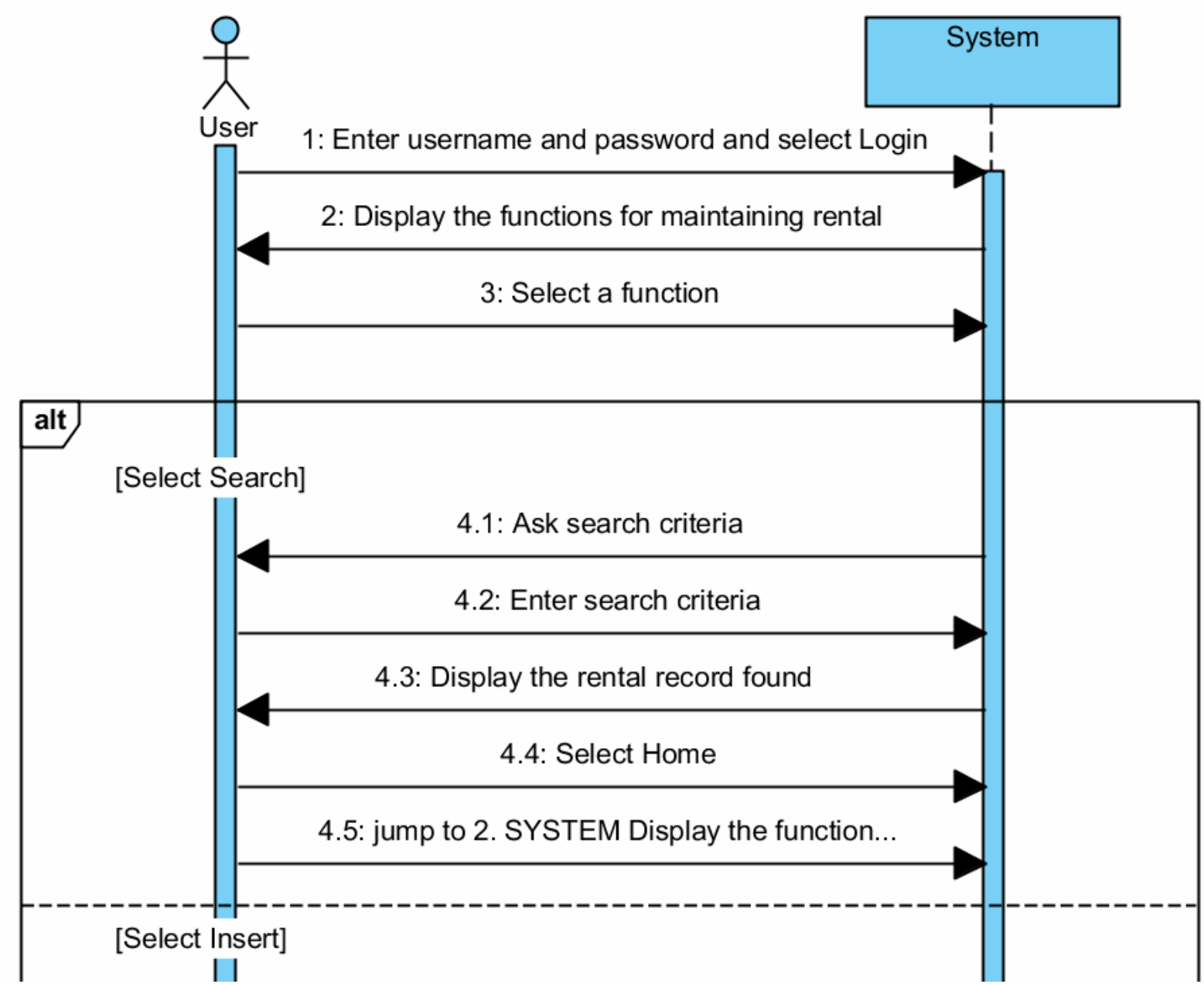
Hide Wireframe

Sequence Diagram

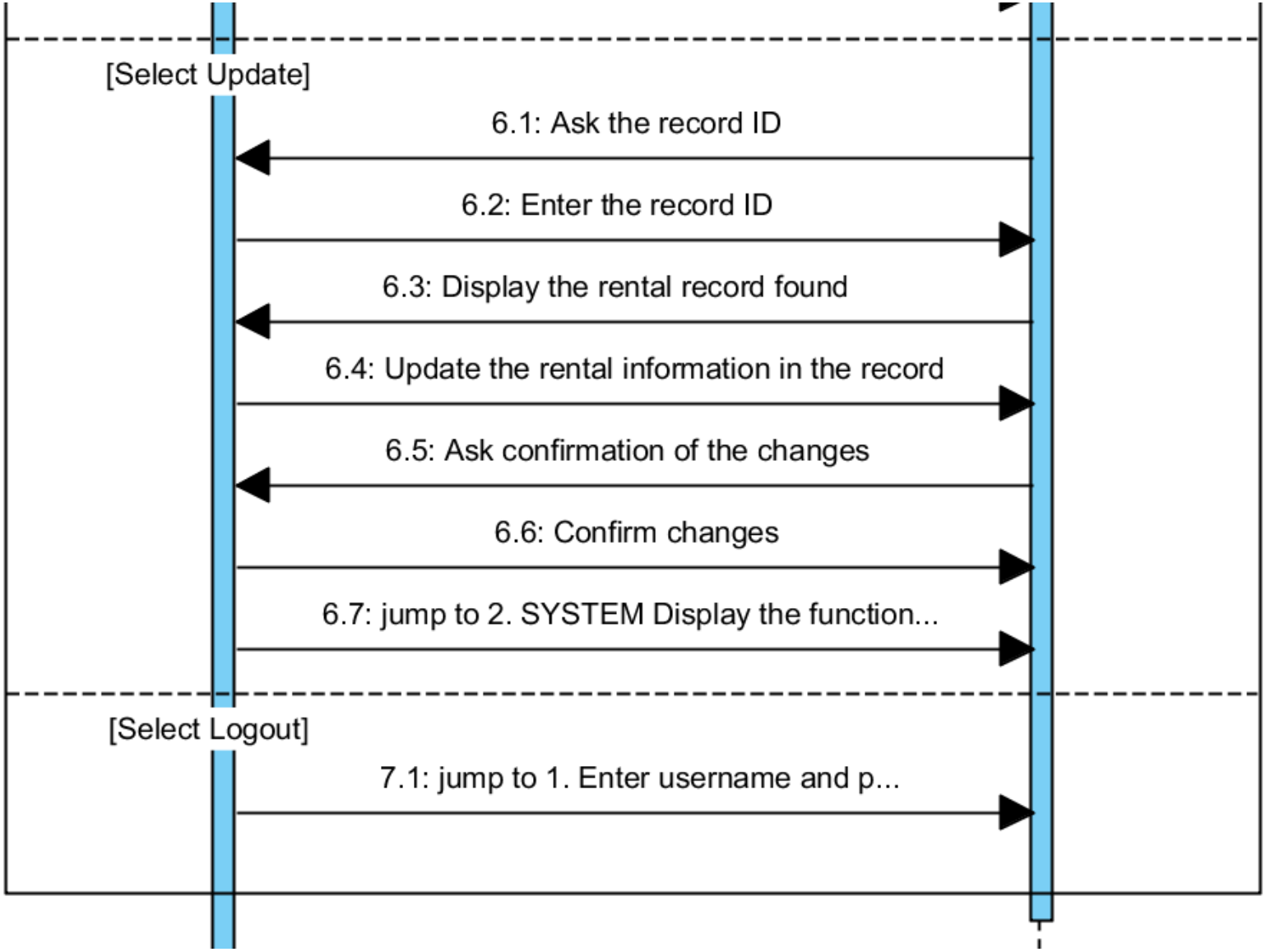
1. Create a sequence diagram related to the *MaintainRental* use case.
2. Tab *Flow of events* → button *Synchronize to Sequence Diagram*



3. The sequence diagram is created and shown.


System
User
1: Enter username and password and select Login
2: Display the functions for maintaining rental
3: Select a function
alt
[Select Search]
4.1: Ask search criteria
4.2: Enter search criteria
4.3: Display the rental record found
4.4: Select Home
4.5: jump to 2. SYSTEM Display the function...
[Select Insert]

. . .


[Select Update]
6.1: Ask the record ID
6.2: Enter the record ID
6.3: Display the rental record found
6.4: Update the rental information in the record
6.5: Ask confirmation of the changes
6.6: Confirm changes
6.7: jump to 2. SYSTEM Display the function...
[Select Logout]
7.1: jump to 1. Enter username and p...

2001, J. Arlow, *UML and the Unified Process*, pages to read during the project:

- from page 63 (Section 4.4.1, only *flow of events*) to page 70 (Section 4.4.4)
- from page 83 (Section 5.3) to page 84; page 85 (Section 5.4), page 87 (Section 5.5)
- from page 105 (Section 7.2) to page 119 (Section 5.5.1)

2005, J. Arlow, *UML and the Unified Process II*, pages to read during the project:

- from page 244 (Section 12.6) to page 263 (Section 12.10.2)