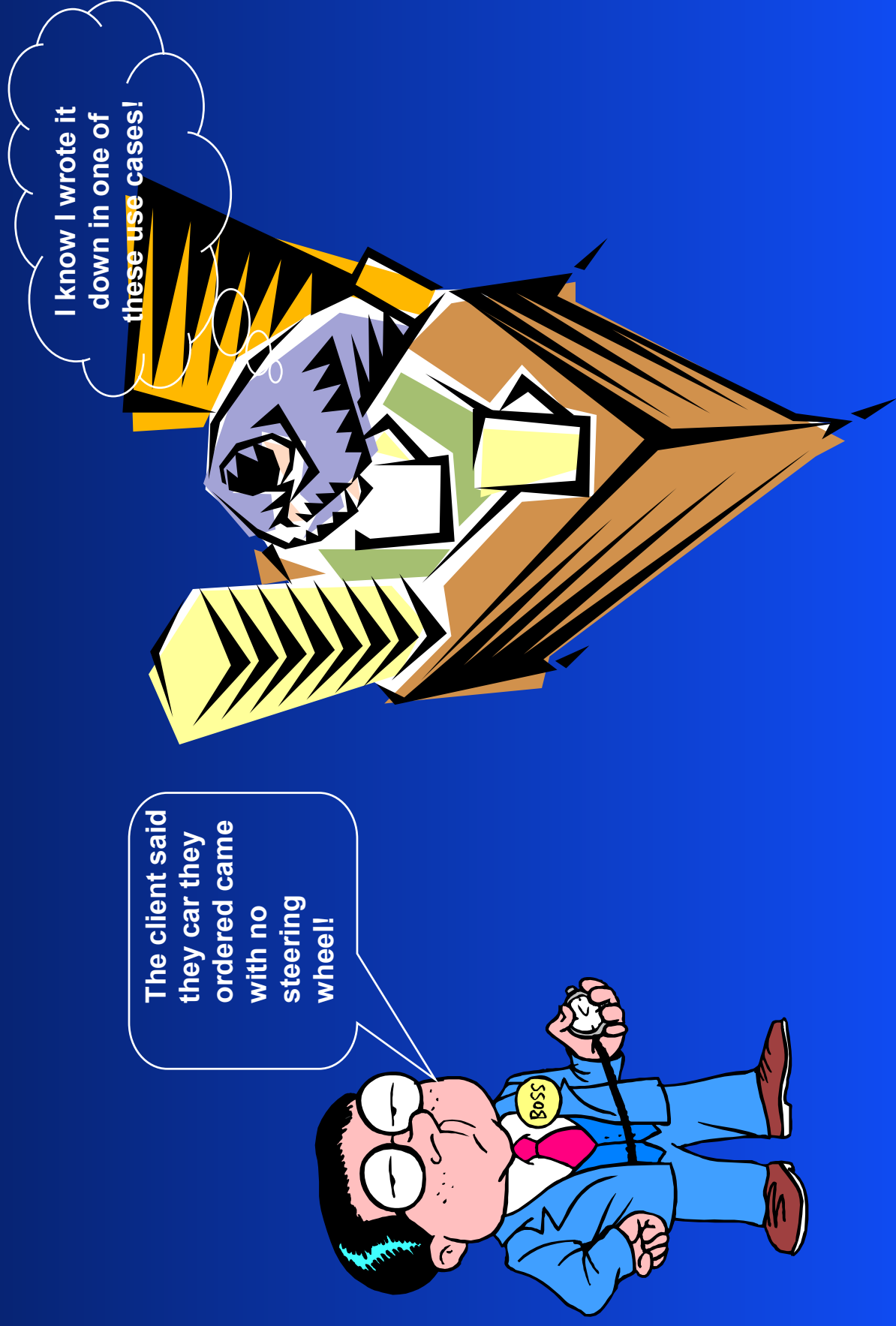


Requirements Engineering with the UML

Brian Berenbach
Siemens Corporate Research
brian.berenbach@scr.siemens.com

Why express requirements graphically?

SIEMENS



Graphical Representation

UML Model

Test Plan

Requirements

ProjectPlan

Oh I press this button here.

Requirement	Priority	Stability
1. Requirement	5	Stable
2. Assign Service Set Map To Payer Or Payer Health Plan	6	Stable
3. Select Expression Name	7	Stable
4. Demand Process Reimbursement Calculations	3	Stable
5. Manage Billing Impact On Receivables	6	Stable
6. Configure Business Office	5	Will Change
7. Configure Account Type Rules	3	Stable
8. Enter Revenue Code Table Rate	7	Stable
9. Enter Fee Schedule Rate	9	Stable
10. Enter Fee Schedule Rate	9	Stable

Topics

- How do I start?
- When am I done?
- How do I create a verifiable model?
- How do I extract my requirement set from the completed model?

Overview of the UML

❑ The UML is a language for

❑ visualizing

❑ Specifying

Use Cases
Business Model
Requirements

Analysis

Design

❑ constructing

❑ documenting

the artifacts of a software-intensive system

Mapping of Diagrams - Logical

Diagram Type	Business Definition
Use Case	A pictorial representation of requirements as viewed from outside the business entity
Sequence	A detailed, step by step depiction of a cross-object business process
Collaboration	Another view of cross-object interaction.
Class	A static representation of the infrastructure of the business, showing the relationship between the various business objects
State	The detailed logic for a single object Service (or class method)
Activity	A specialized State diagram that can show concurrent activity.

Mapping of Diagrams - Physical SIEMENS

Diagram Type

Business Definition

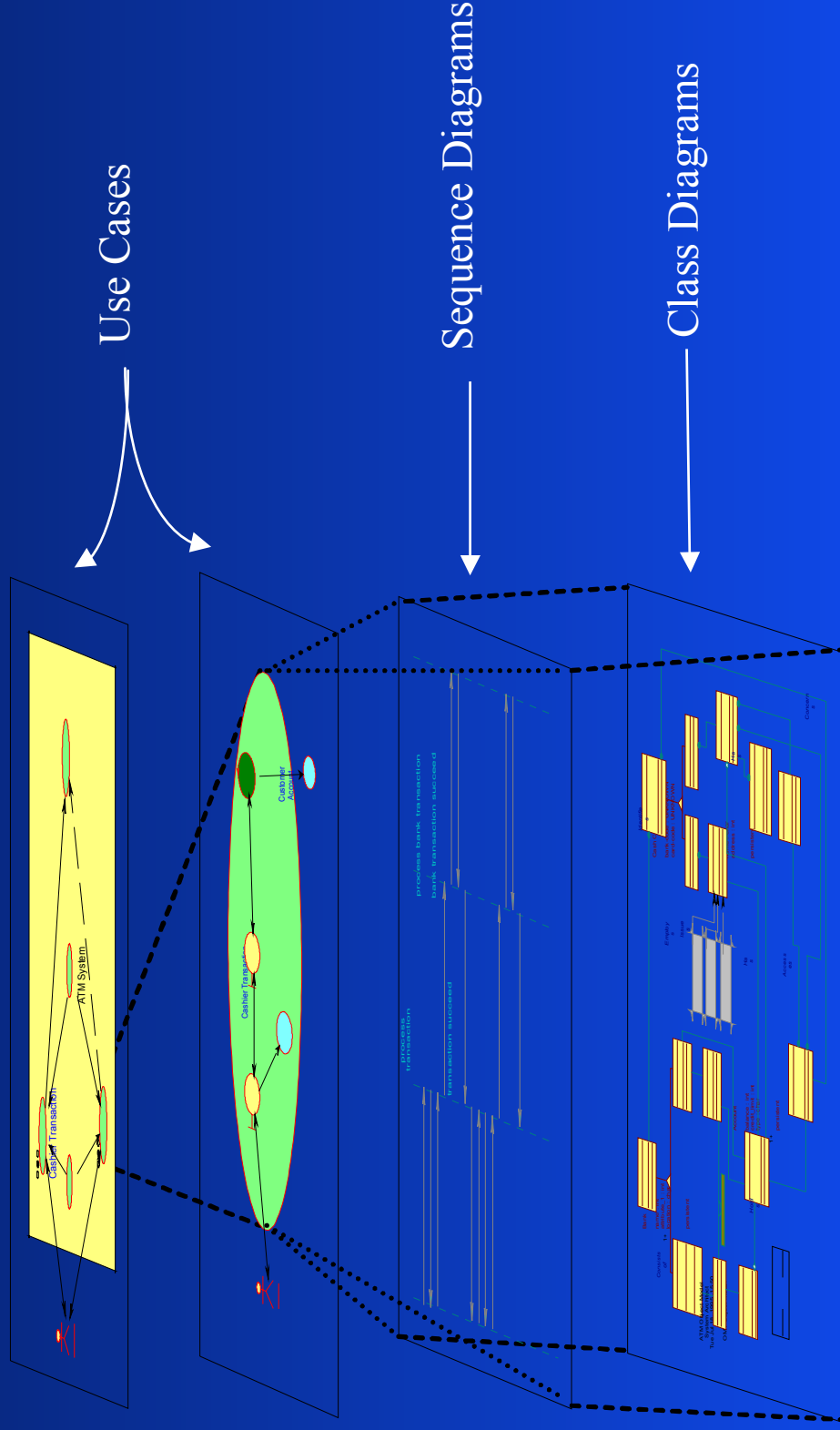
Component

A diagram showing physical software components and their interactions through defined interfaces.

Deployment

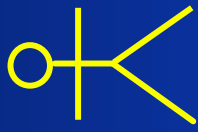
A diagram showing the deployment of software on nodes.

Modeling Topology



The Use Case Diagram

Use Case Symbols



External to the
business system

An Actor



A business object that controls
the flow of work

A Control Object



A business
process

A Use Case



Any object manipulated by a
business process

Ordinary Object



A business Object
that communicates
with actors

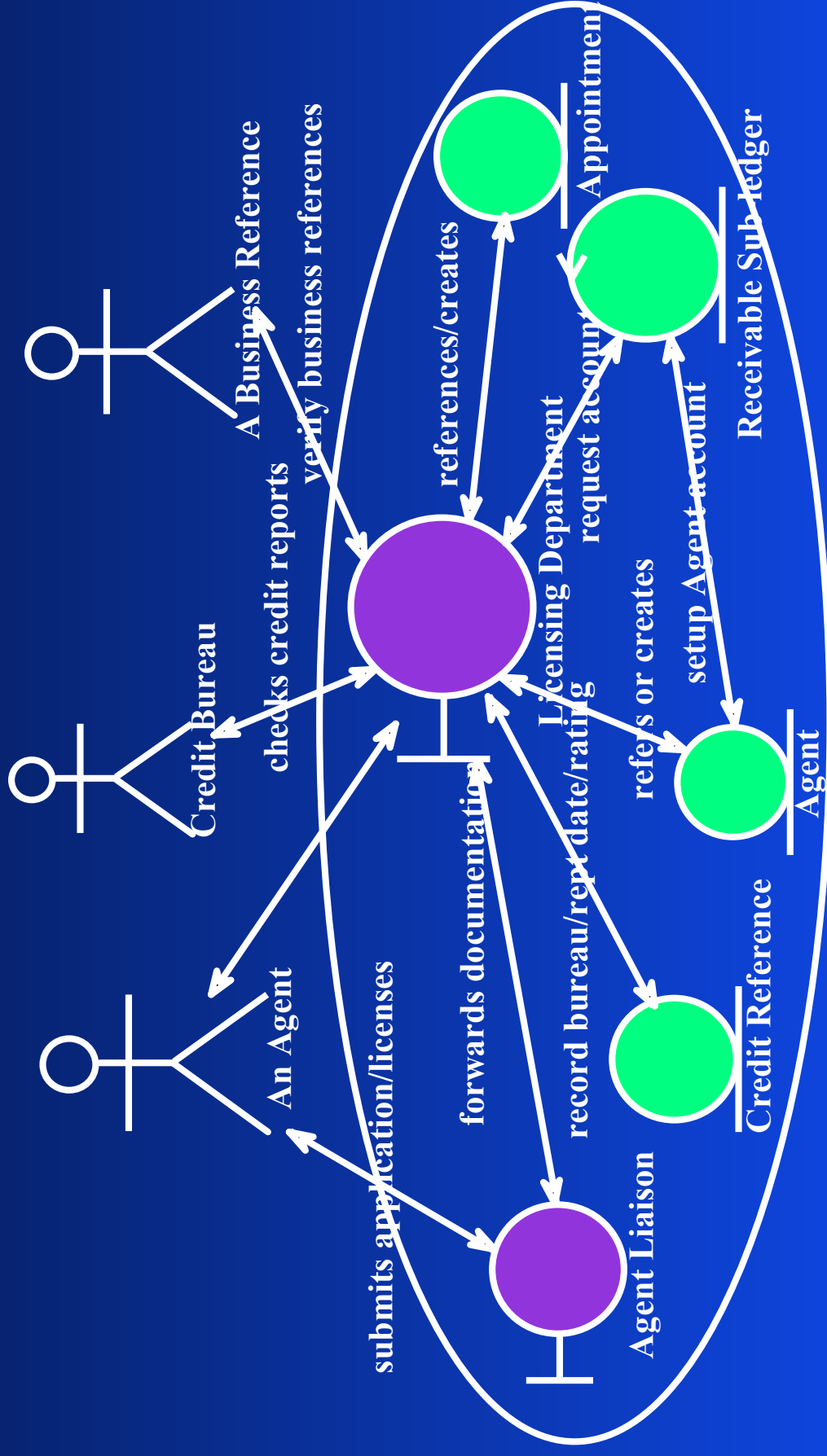
An Interface



An identifiable part of a business that
contains business processes.

A Business System

Sample Use Case-Insurance

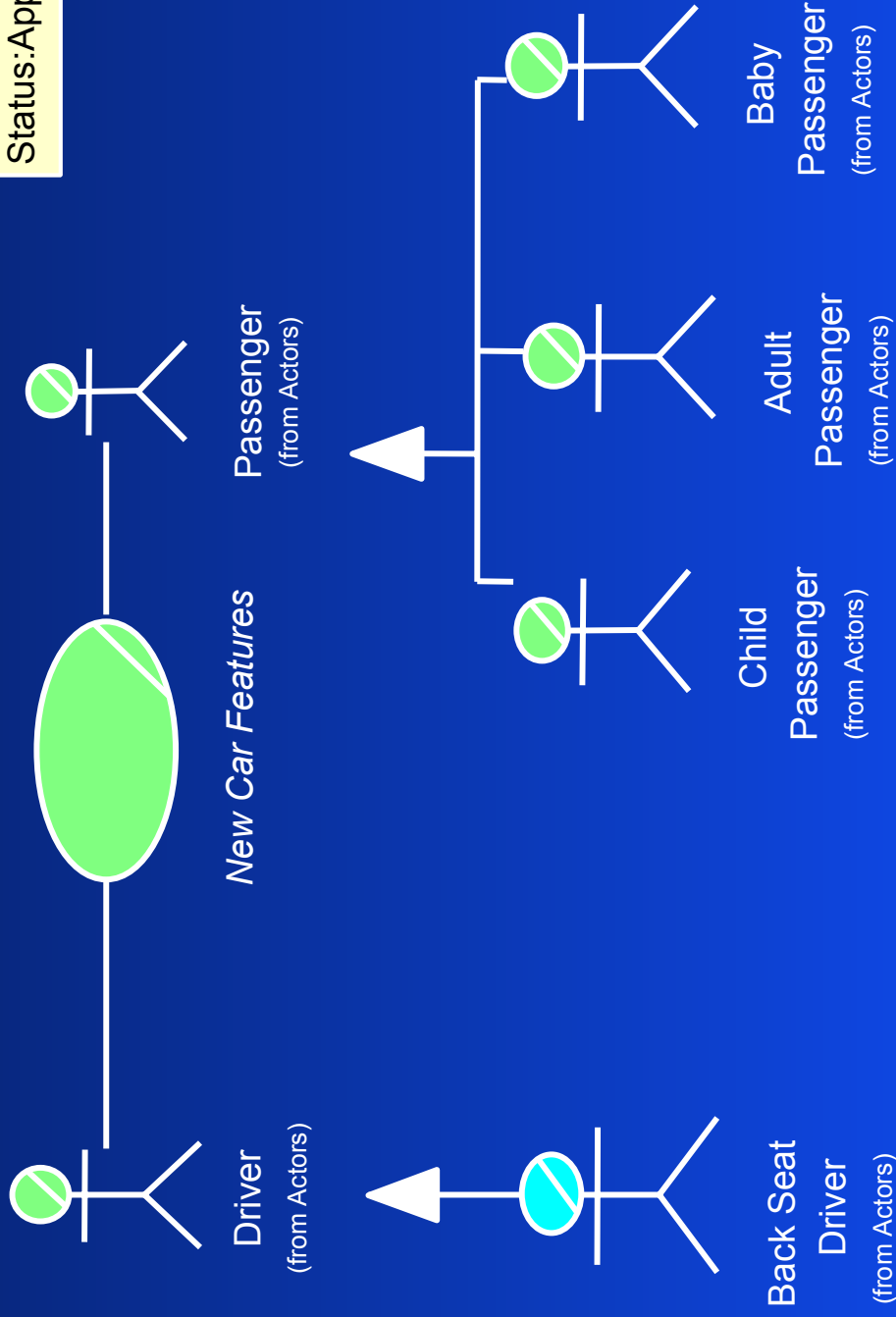


Appoint Agent

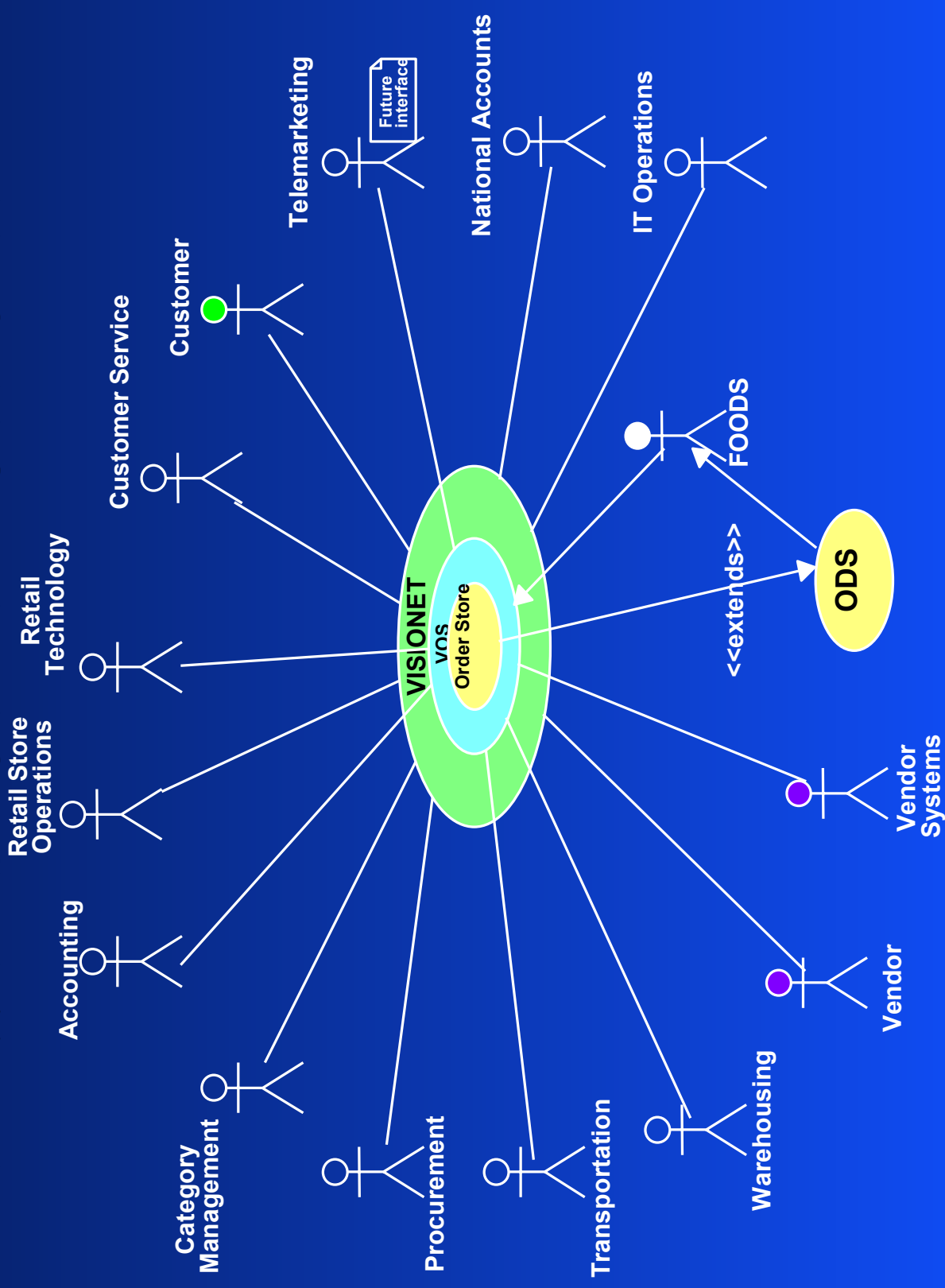
Problem: Discover the features and options associated with buying a new car, then get all the requirements needed for power windows.

- The model should have a single entry point.
- All the actors in the model should be shown on the “context” diagram

Context Diagram
 Creation Date: 2/26/2003
 Modification Date: 2/26/2003
 Modified By: brian
 Status: Approved

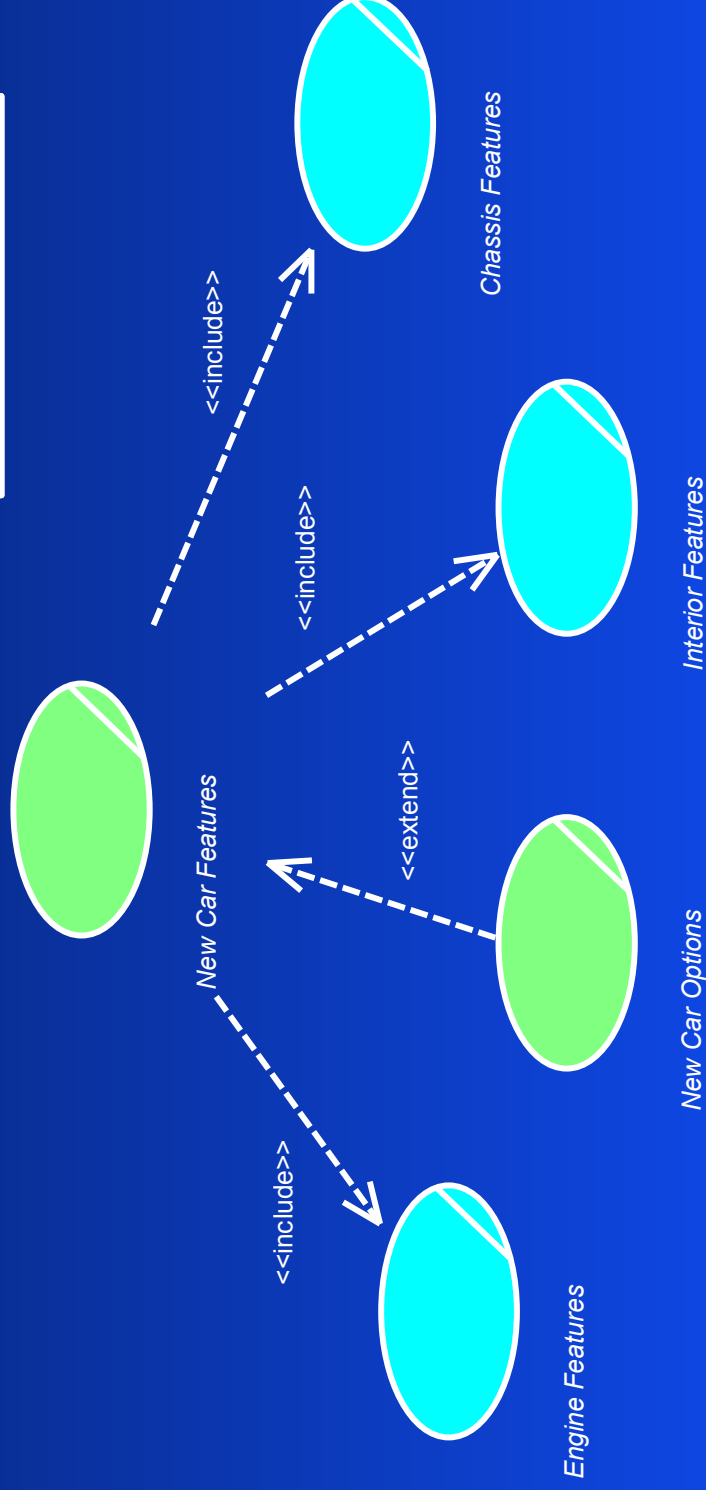


“Real World” Context Use Case



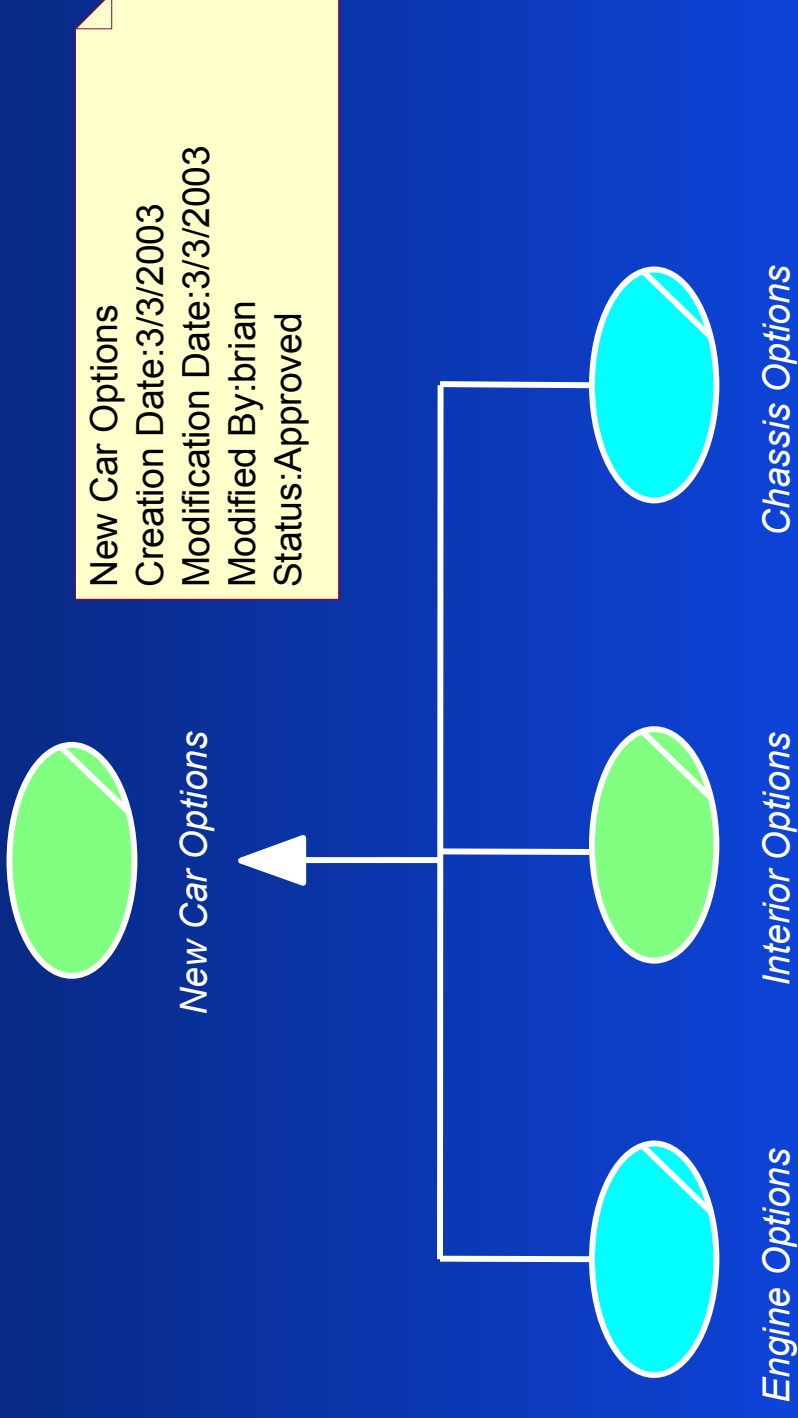
- The early modeling effort should cover the entire breadth of the domain.
- Identify “out of scope” use cases as early as possible

New Car Features
 Creation Date: 3/3/2003
 Modification Date: 3/3/2003
 Modified By: brian
 Status: Approved



Relationships

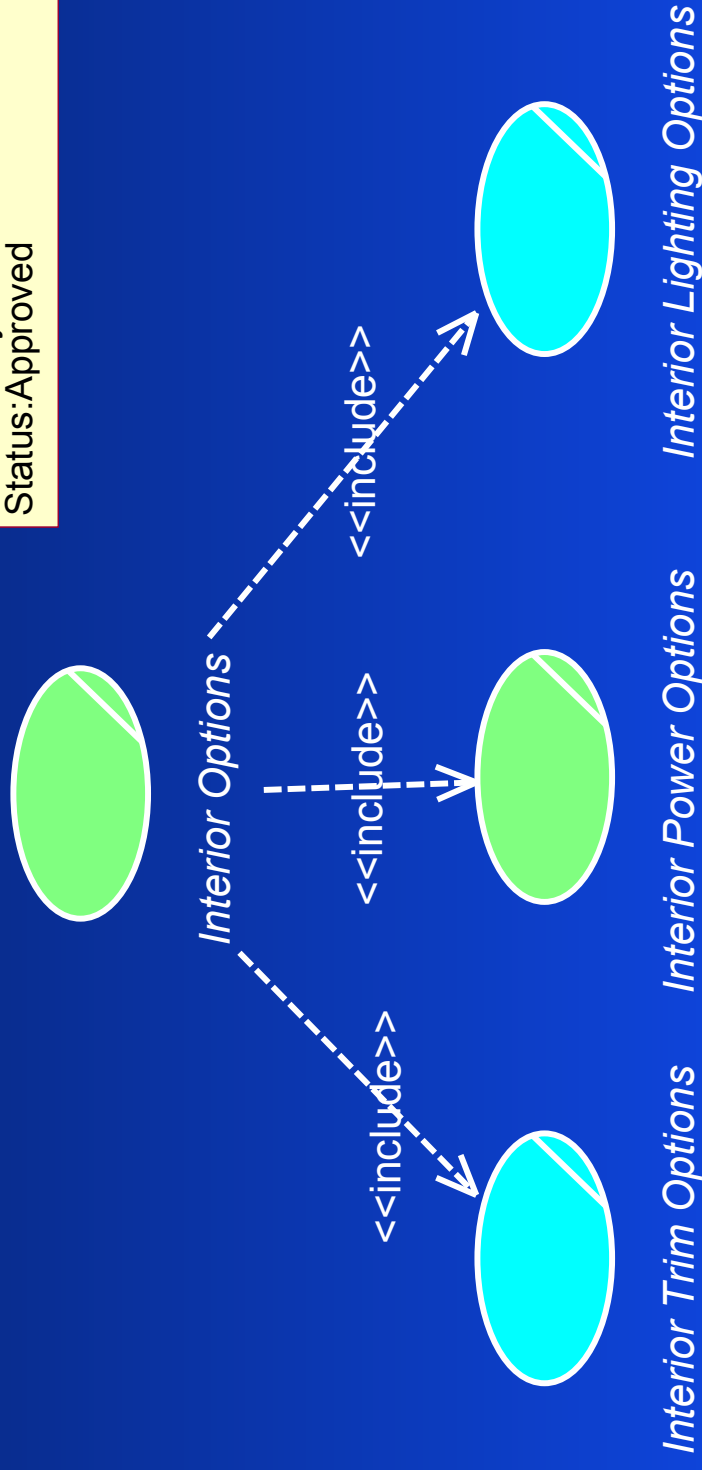
- *When a use case includes another use case, the included use case is an integral part of the including process.*
- *When a use case extends another use case, the extending use case is optionally part of the process it extends.*



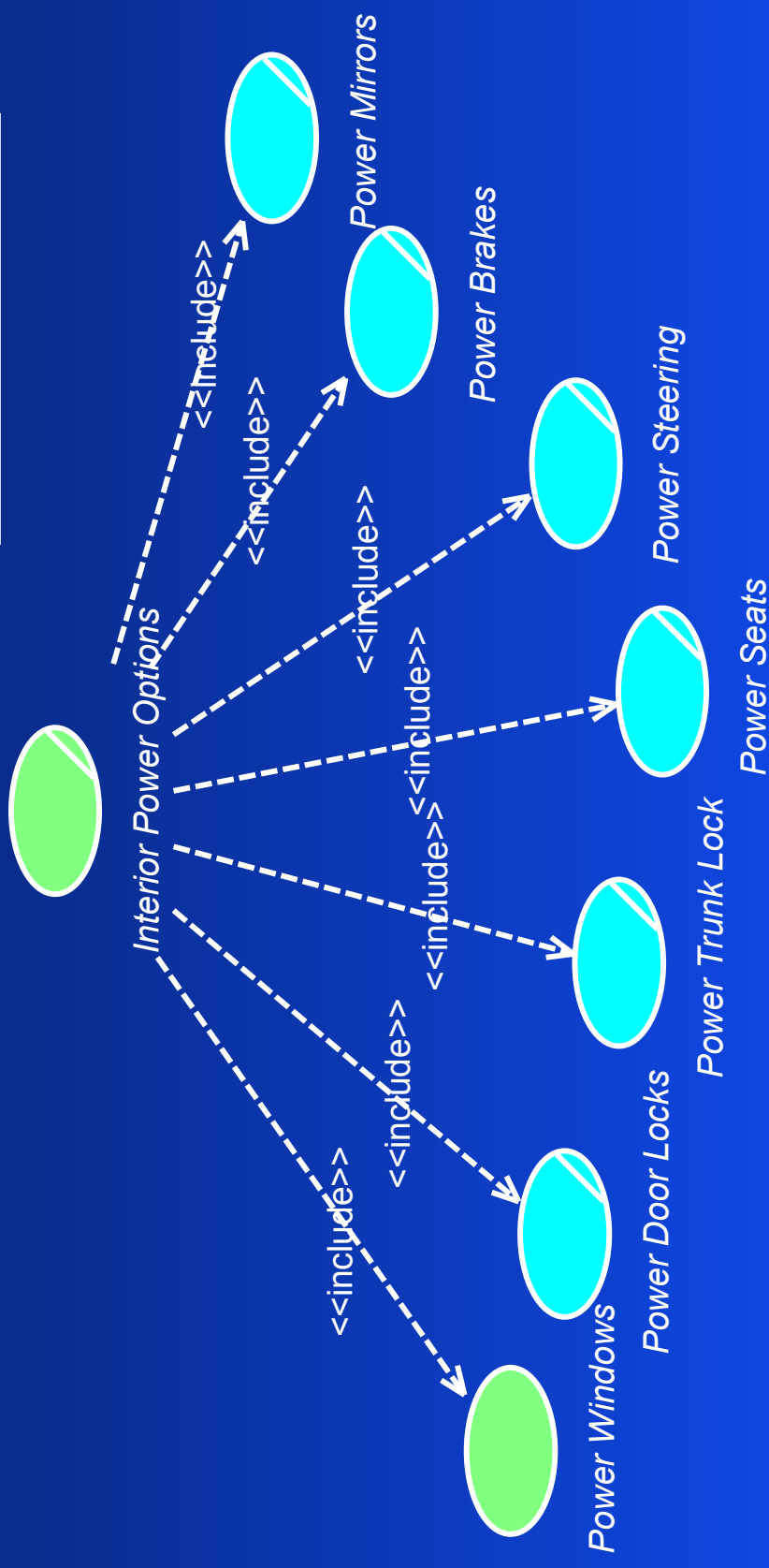
Inheritance

➤ *Inheritance is a “kind of”. When B inherits from A, B is a kind of A*

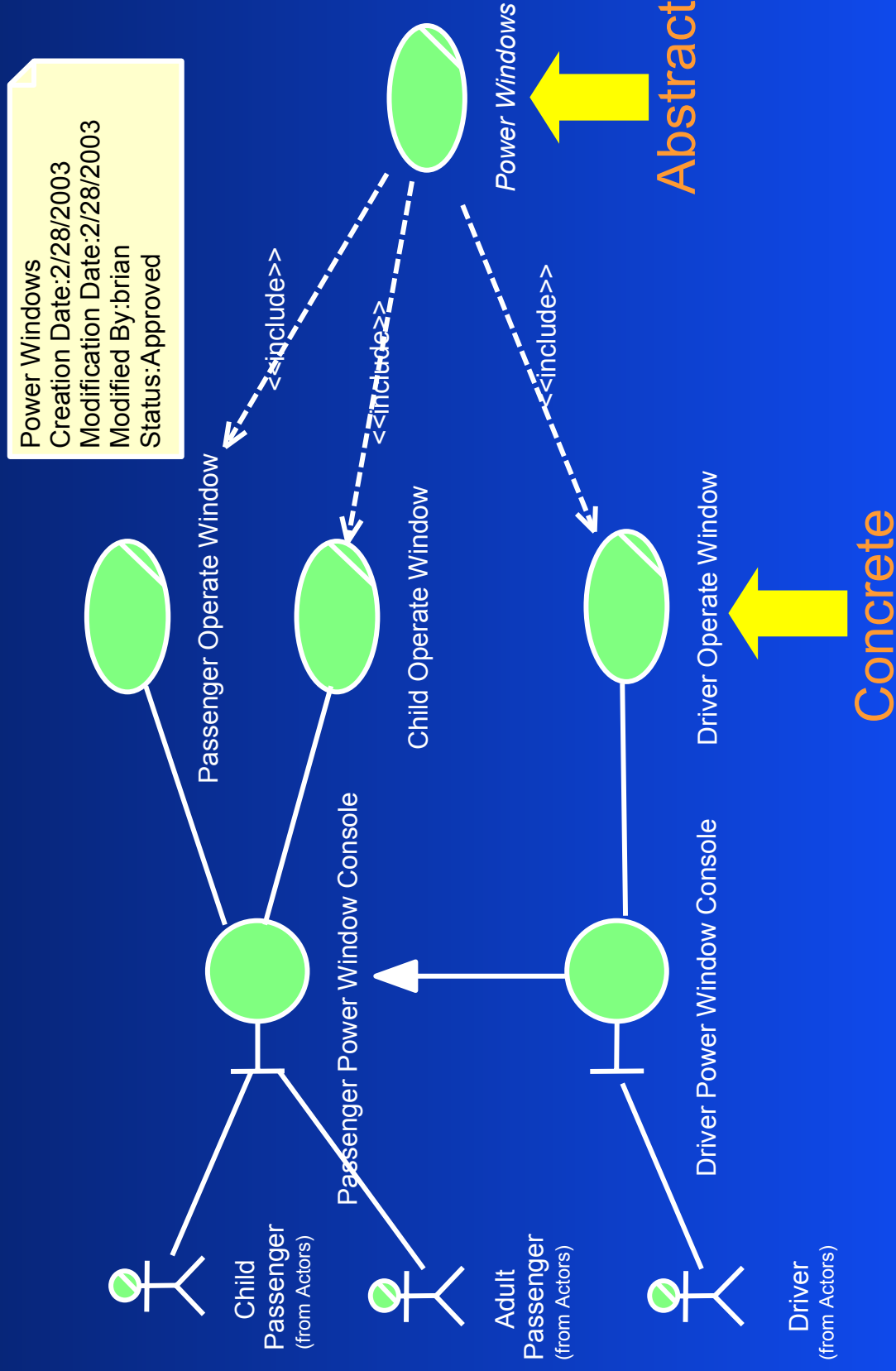
Interior Options
Creation Date:2/26/2003
Modification Date:2/26/2003
Modified By:brian
Status:Approved



Interior Power Options
Creation Date:2/26/2003
Modification Date:2/26/2003
Modified By:brian
Status:Approved



➤ Every actor in the model should communicate with use cases through interfaces



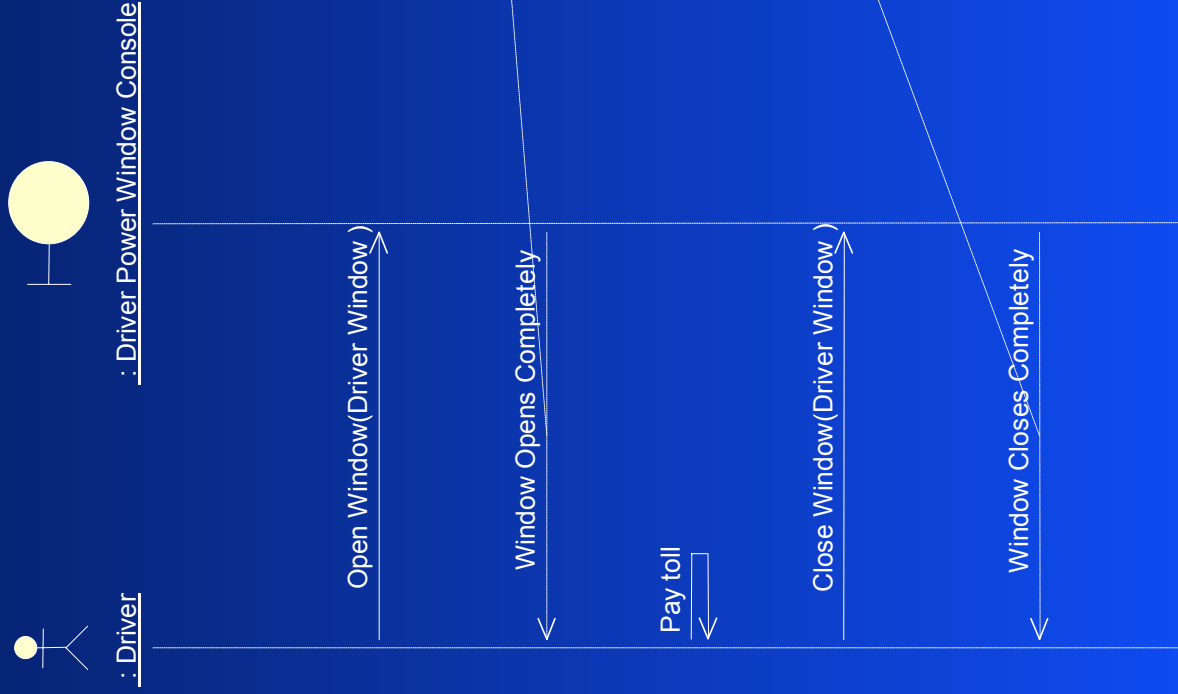
Use case properties

- An abstract use case will never be *realized*. It is a placeholder, an organizer.
- A concrete use case will be *realized* as a real artifact or property of the delivered product.

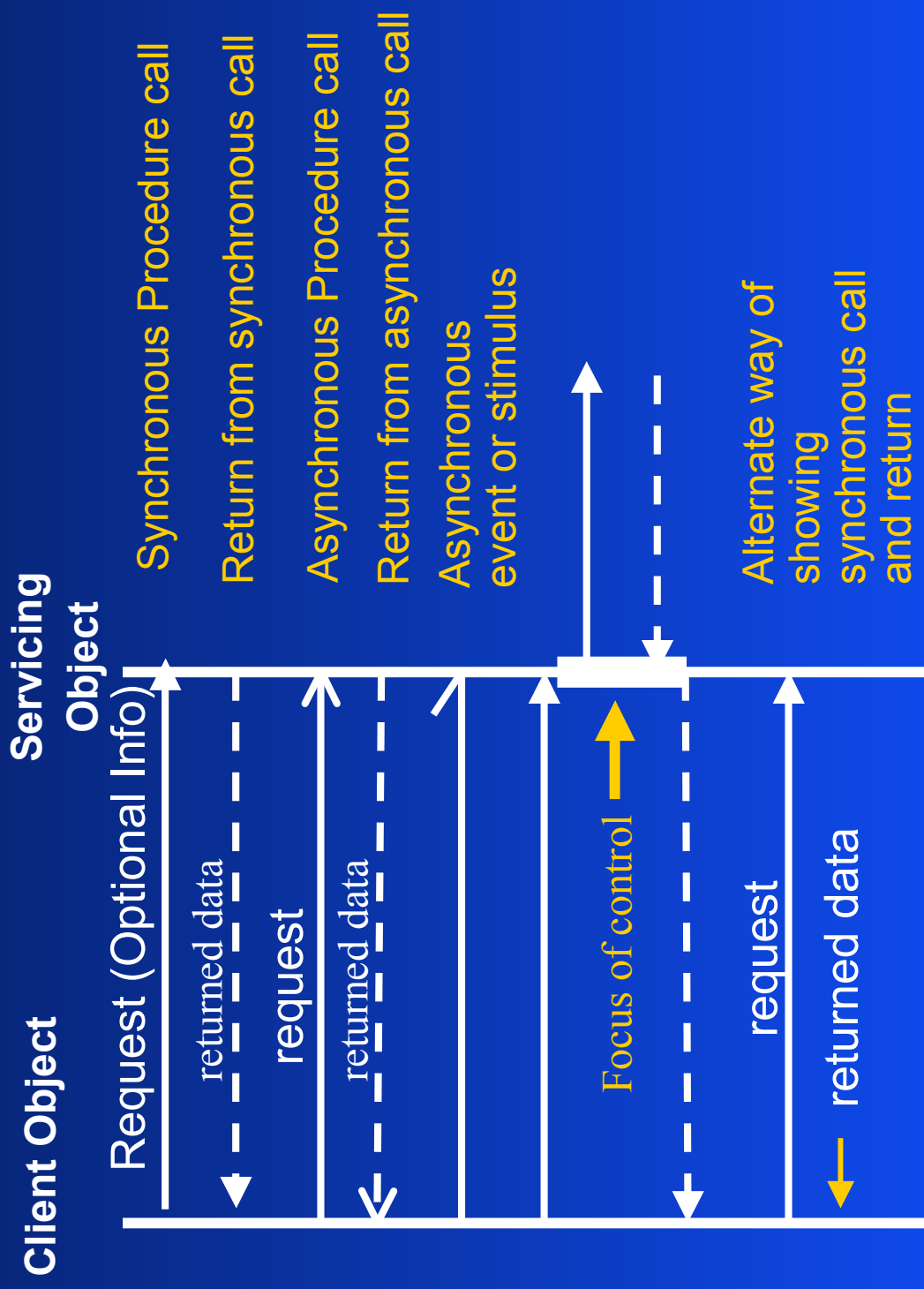
- **Every concrete use case must be defined:**
- **Non-Leaf use cases are defined by sequence, collaboration, activity diagrams**
- **Leaf use cases are services and are defined by activity or state diagrams.**

Use sequence diagrams to define one thread/path for a process

Driver Pay Toll
 Creation Date:2/28/2003
 Modification Date:2/28/2003
 Modified By:brian
 Status:Approved



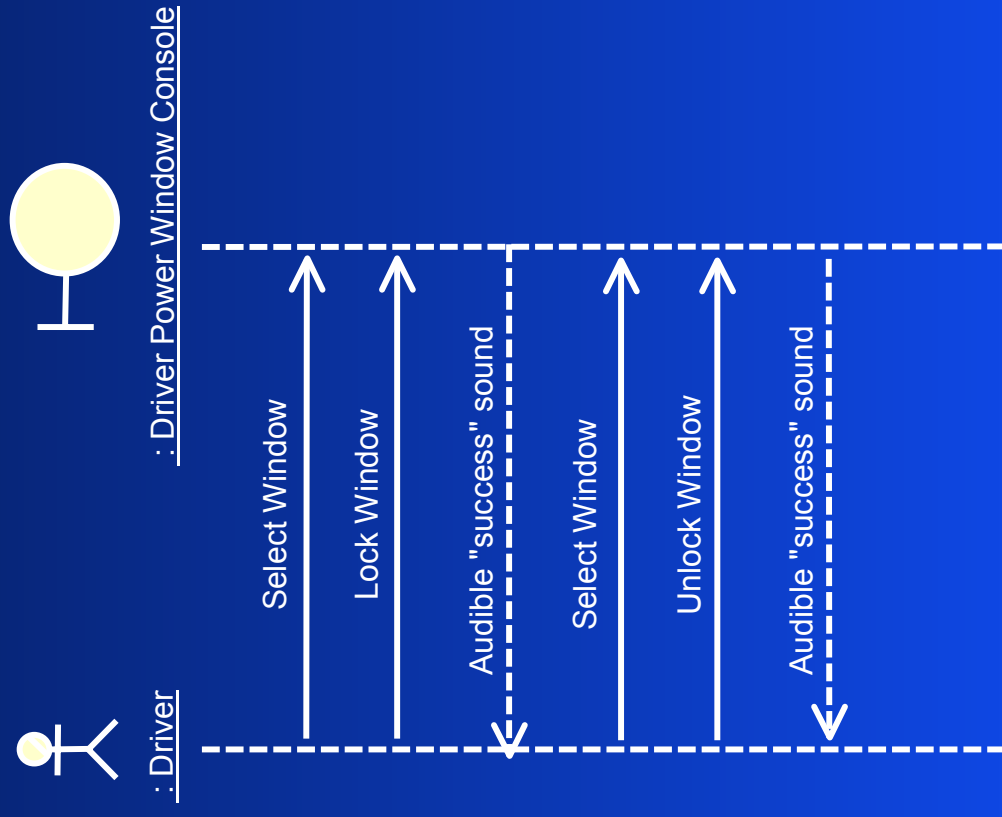
Sequence Diagram Notation



Discover business objects and their services through use case definition with sequence diagrams.

Driver Power Window Console	
	Open Window
	Close Window

Driver Lock or Unlock Window
Creation Date:2/28/2003
Modification Date:2/28/2003
Modified By:brian
Status:Approved

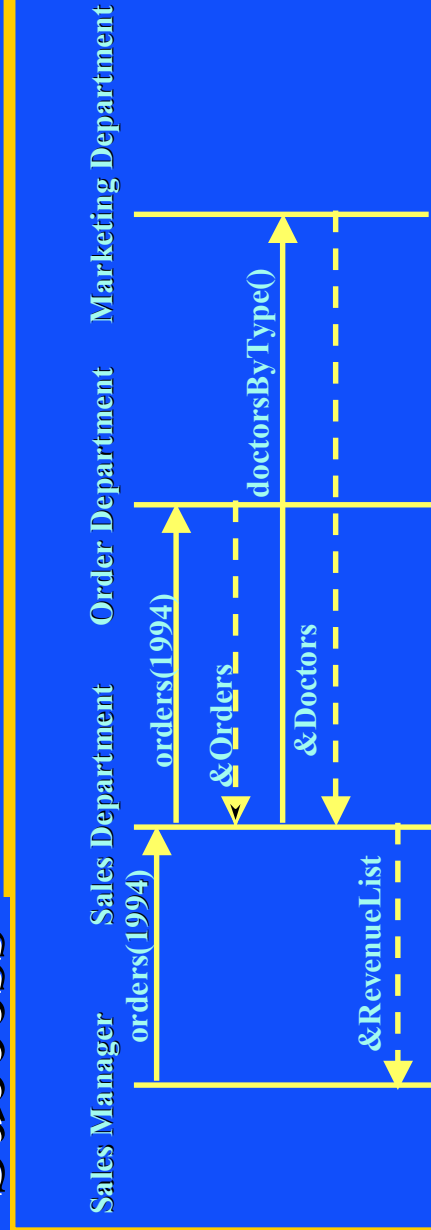


Completing sequences

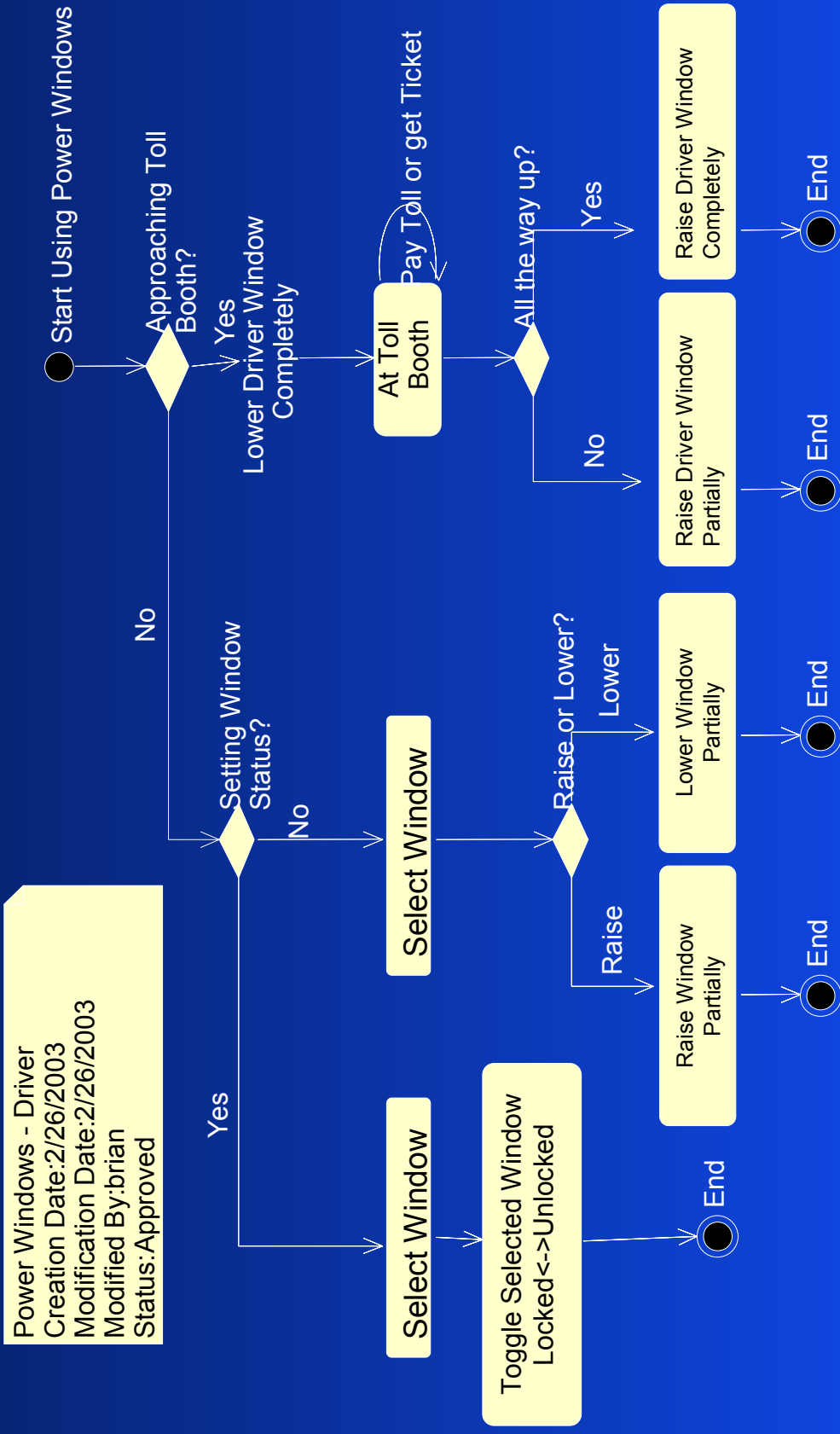
Failure 2

Failure 1

Success



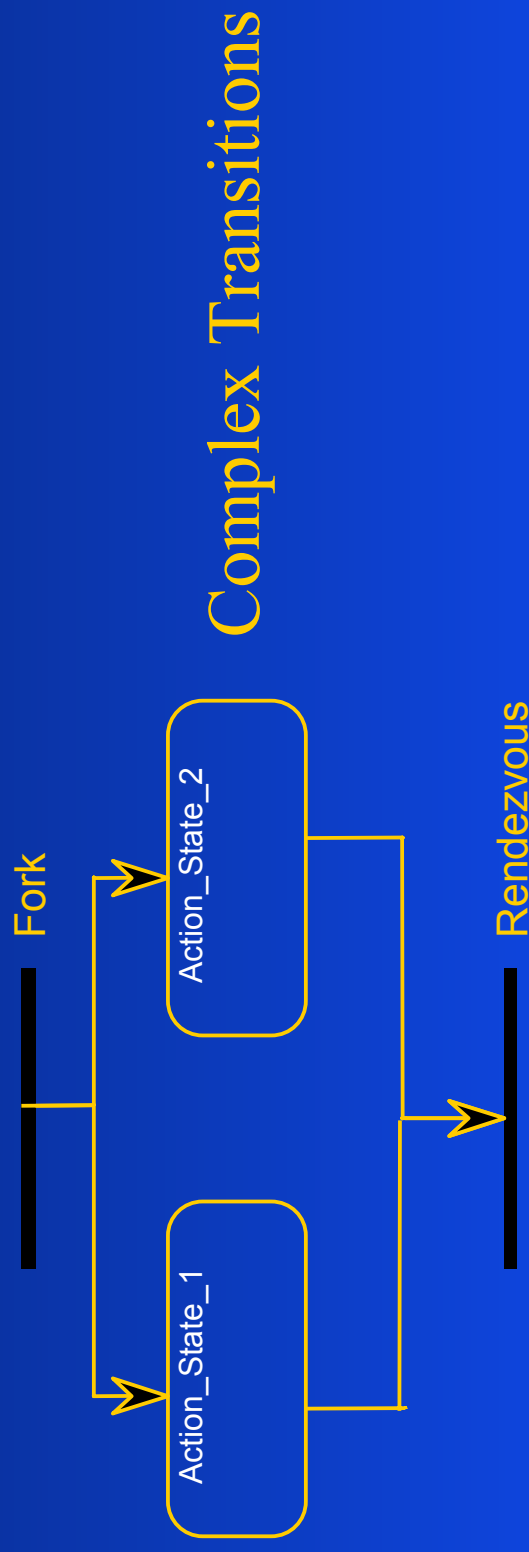
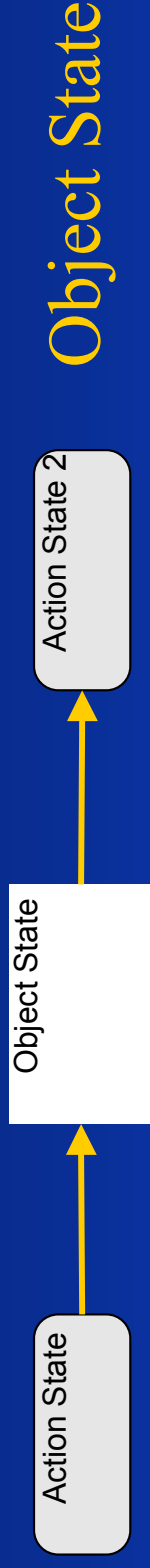
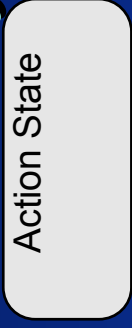
**Use an activity diagram to show
all possible scenarios associated
with a use case**



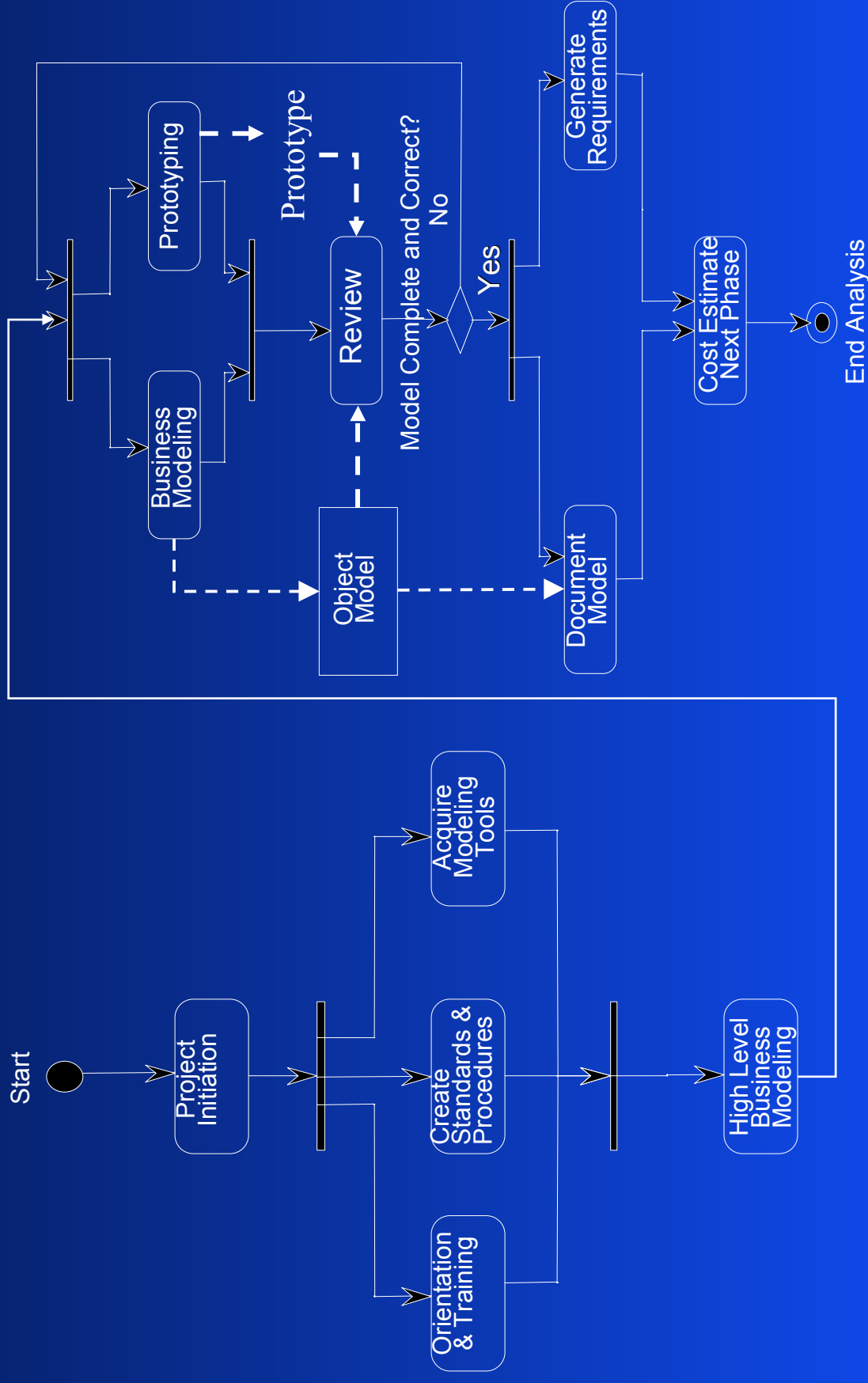
The Activity Diagram

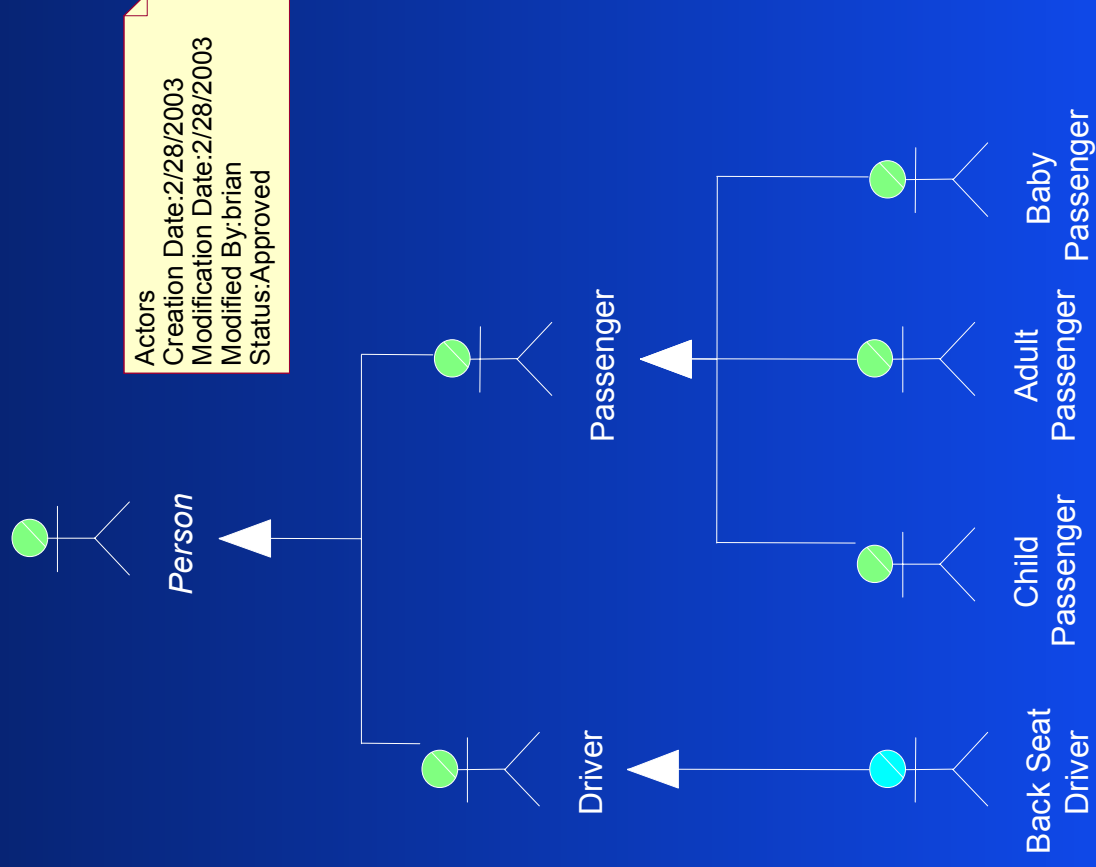
Activity Diagram Notation

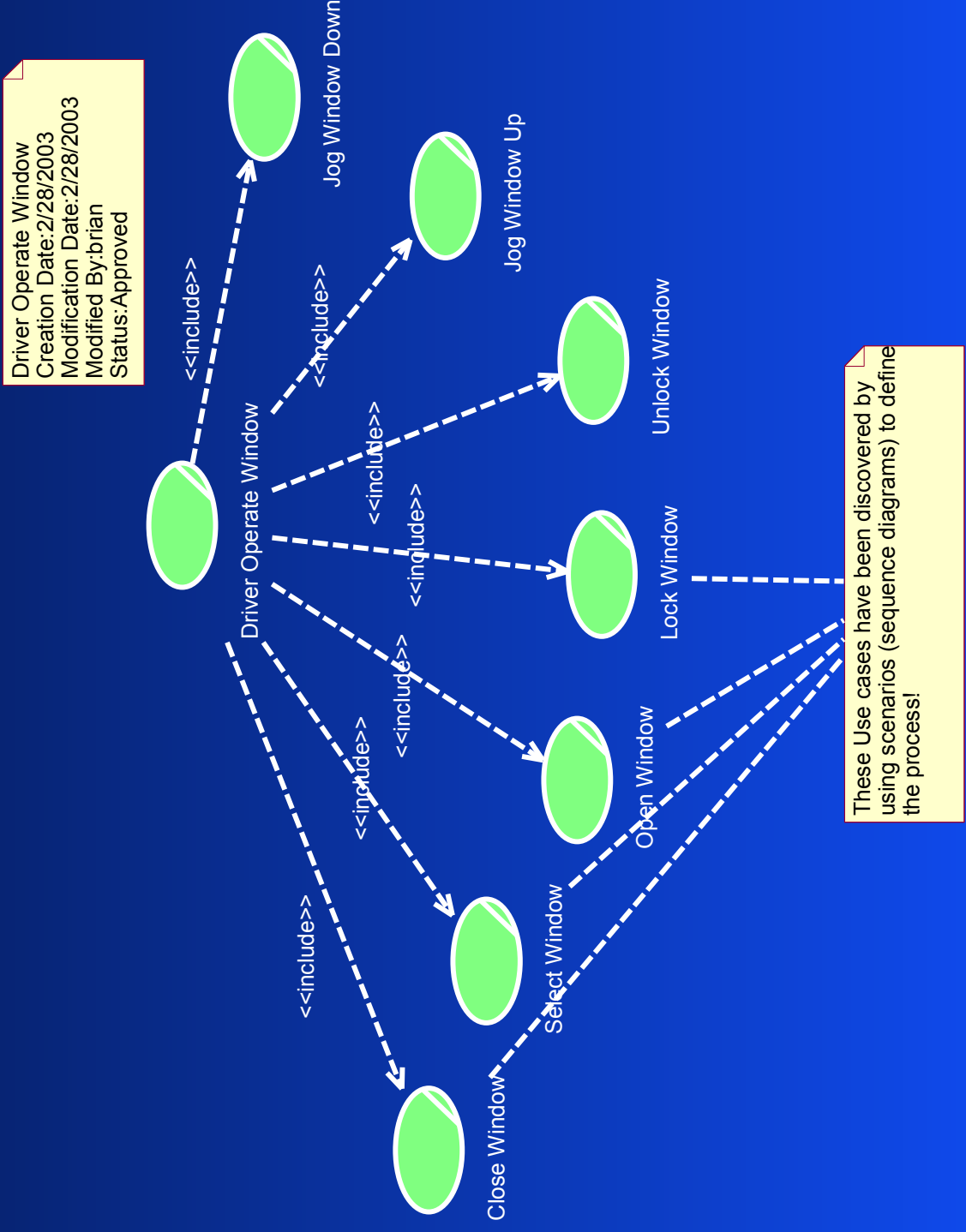
Action State



Analysis Summary



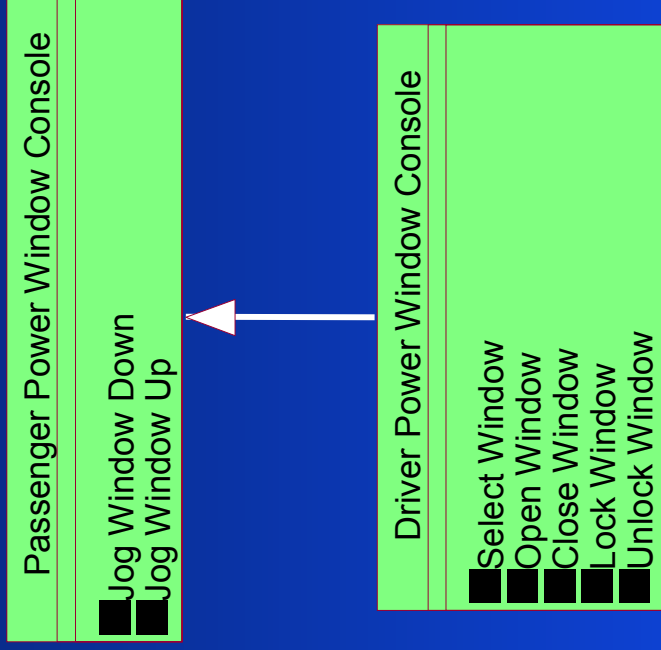




Identification of Services



Power Window Consoles
 Creation Date:2/28/2003
 Modification Date:2/28/2003
 Modified By:brian
 Status:Approved



Customer Object Services

Customer

Accumulate Quantities (Sequenced Items)

This service accumulates the various quantities for the same item to get a sub total prior to applying the netting algorithm.

Confirmation Number

Once an order has been scheduled , the confirmation number is returned to the requesting individual or client for use as a tracking number.

Create

This is the service for creating a new object of type customer.

Get

This service returns customer information

Get Credit Status

This service returns a customers current credit status

Is Valid (Retail Department)

Determine if the specified retail department is valid.

Net Quantities (Customer , Delivery Date , Item , Run , Department)

Net Order Quantities is the process of combining all duplicate items within an order for purposes of validating the total quantity being ordered for a single item. Stores may order the same item multiple times within a single order.

Return Display

System Response to the user initiated "View By Customer Order" service is to render the constructed form onto the User's display and permit the user to interact with the form.

Uses (Retail Department)

Determine if the specified customer uses this retail department.

Uses Retail Department?

Determine if a customer uses a specific retail department.

Validate (Credit Status)

This service validates a customer's credit status.

Validate Quantity (Item Number , Accumulated Quantity)

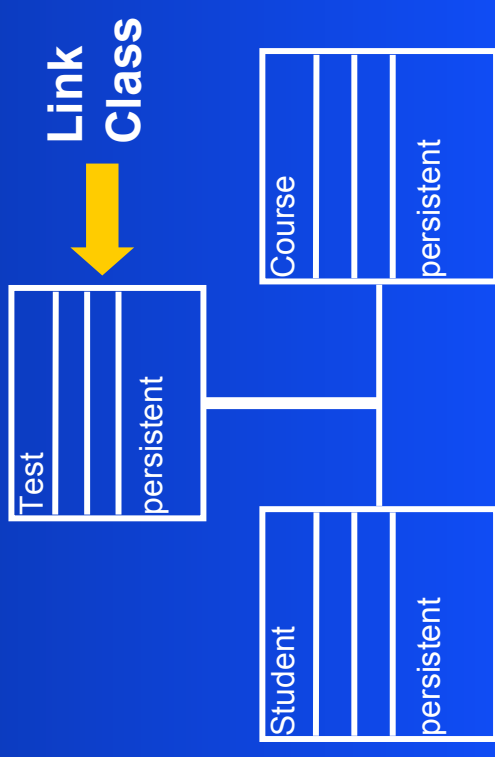
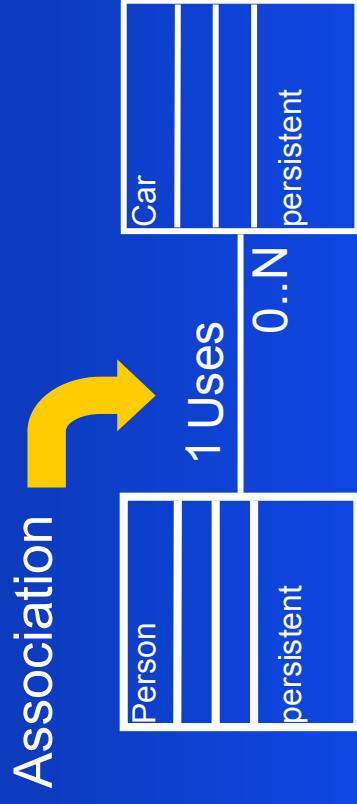
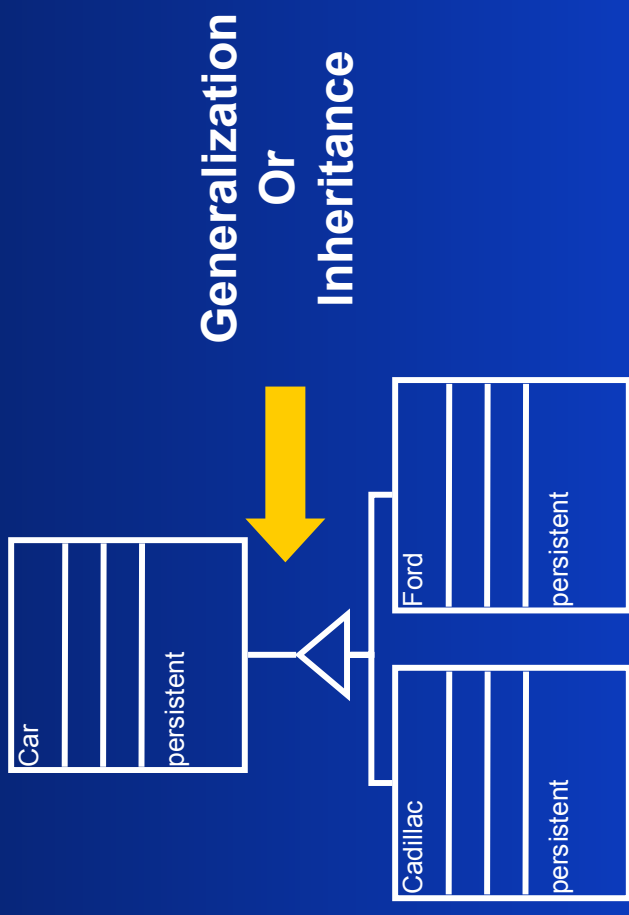
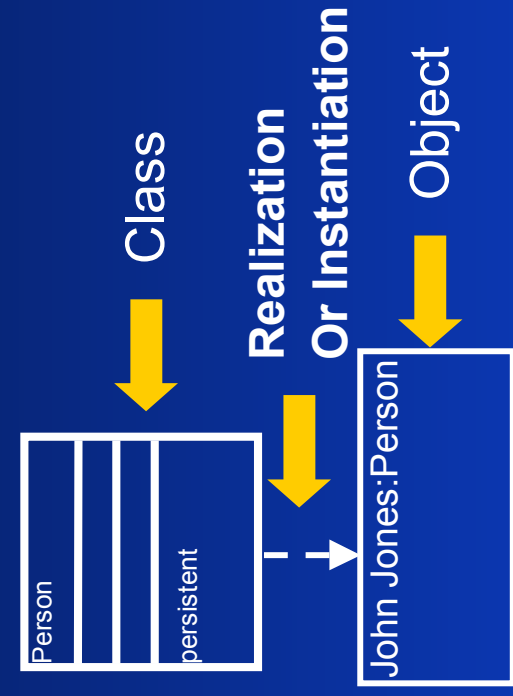
The totaled quantity to be netted must first be validated to confirm that it is a valid quantity.

The Class Diagram

The Class Diagram

- ❑ Each class is a container
 - ❑ Lists object services from sequence diagrams
- ❑ Used also to
 - ❑ Hold key attribute information (data)
 - ❑ Show relationships between objects
 - ❑ Create a persistent store
 - ❑ Generate Software
 - ❑ Software Repository

Class Diagram Symbols



The Class is a Container

<<actor>> My Class	Partition 1 – name and stereotype
Attribute 1 Attribute 2 Attribute 3	Partition 2 – Data
Function "Service 1" as File Function "Service 2" as Account Function "Service 3" as Name	Partition 3 - Services
persistent	Partition 4 – Misc.

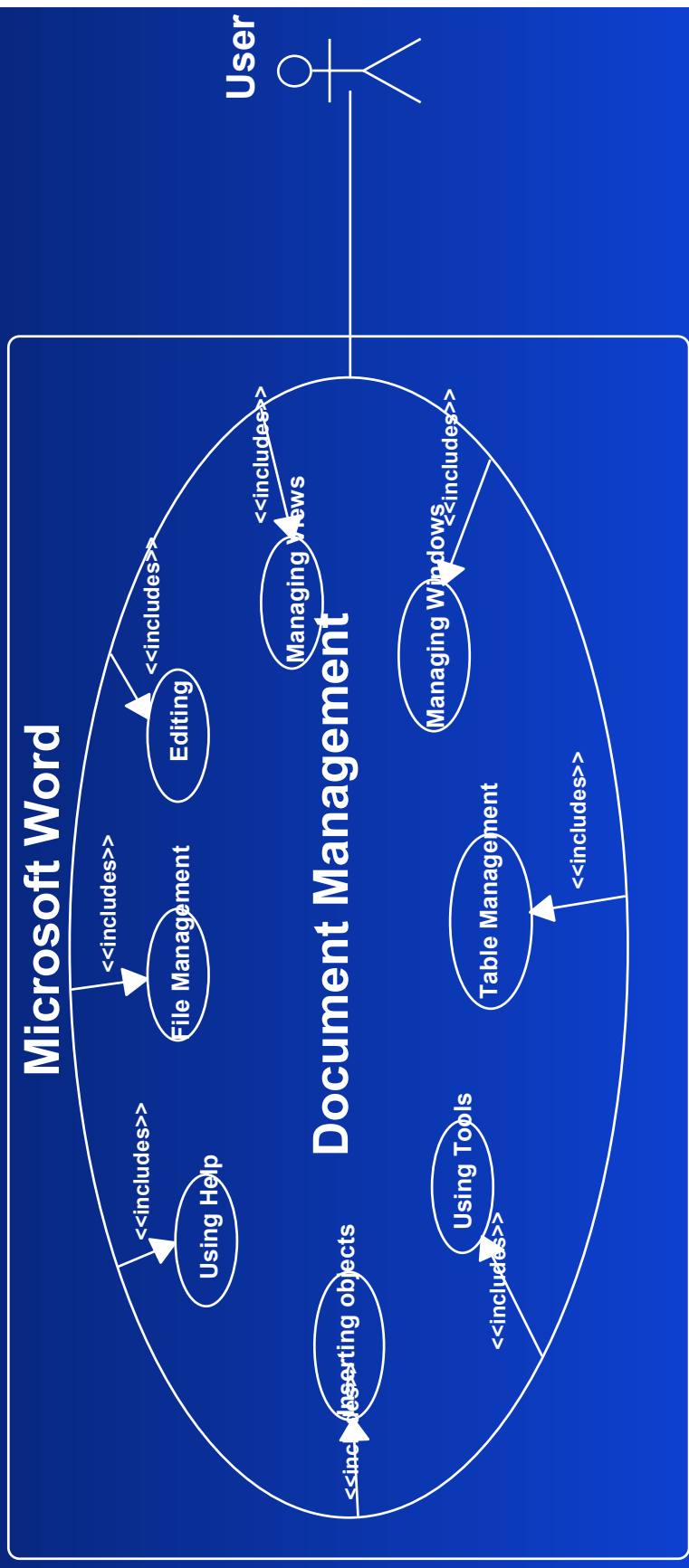
Requirements Extracted from Model

<input type="checkbox"/> New Car Options
<input type="checkbox"/> Interior Options
Interior Trim Options
<input type="checkbox"/> Interior Power Options
<input type="checkbox"/> Power Windows
Passenger Operate Window
Child Operate Window
<input type="checkbox"/> Driver Operate Window
Close Window
Select Window
Open Window
Lock Window
Unlock Window
Jog Window Up
Jog Window Down
Power Door Locks
Power Steering
Power Brakes
Power Mirrors
Power Trunk Lock
Power Seats
Interior Lighting Options

Prototyping

- ❑ Aid in finding requirements
- ❑ First cut at user interface
- ❑ Helpful in visualizing final results

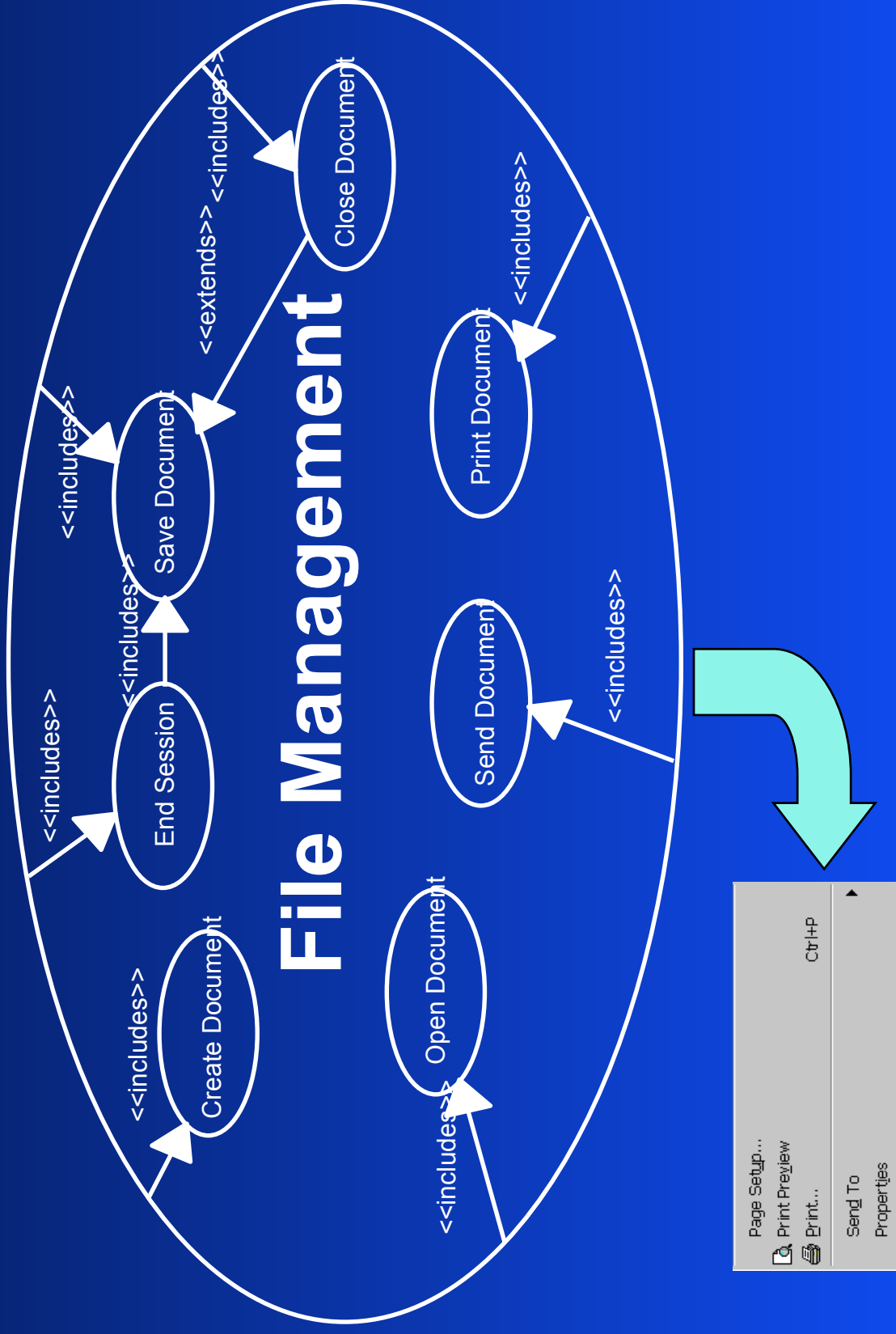
Context use case defines highest level



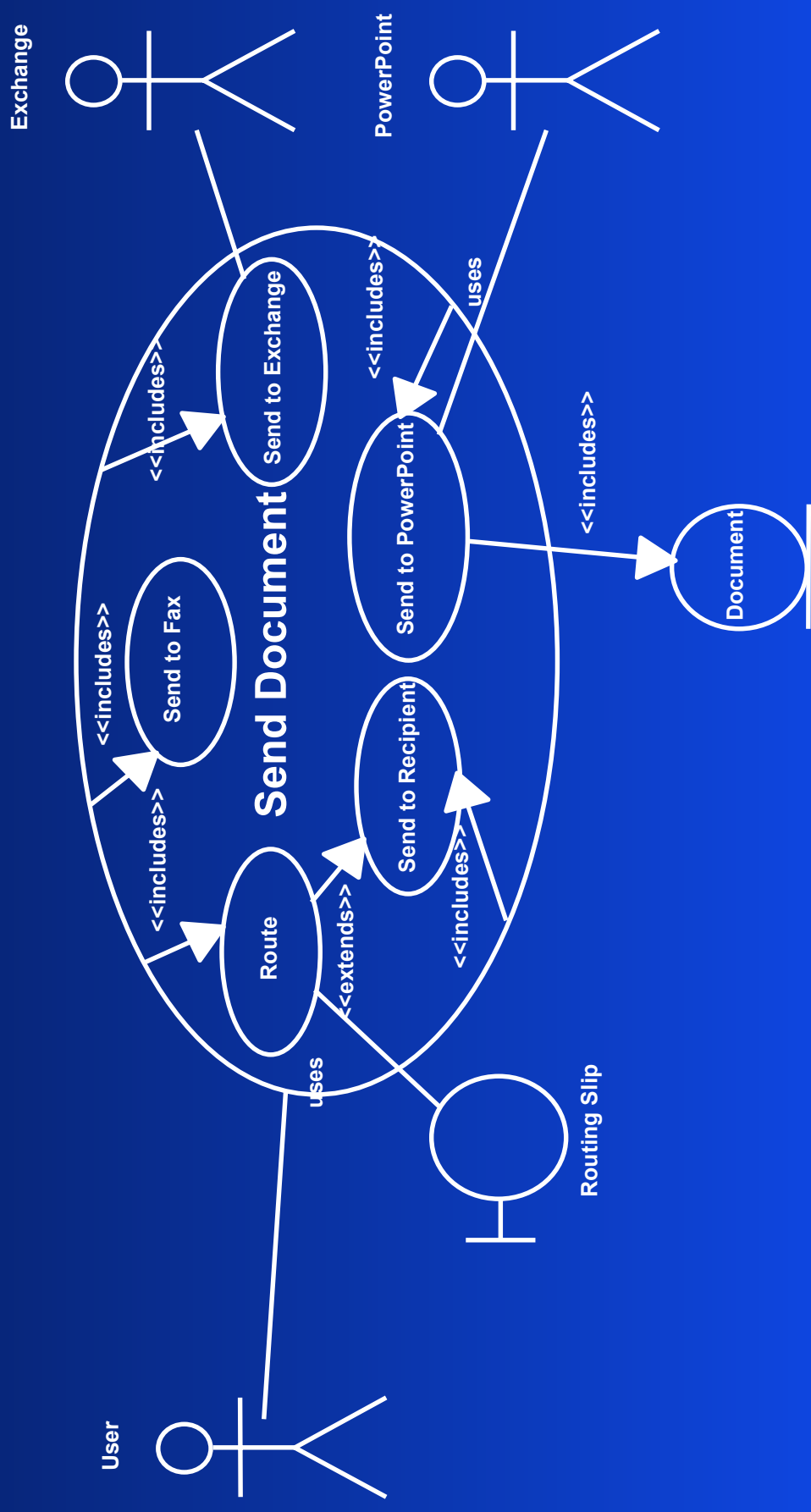
Microsoft Word - UML Prototype.doc

File Edit View Insert Format Tools Table Window Help

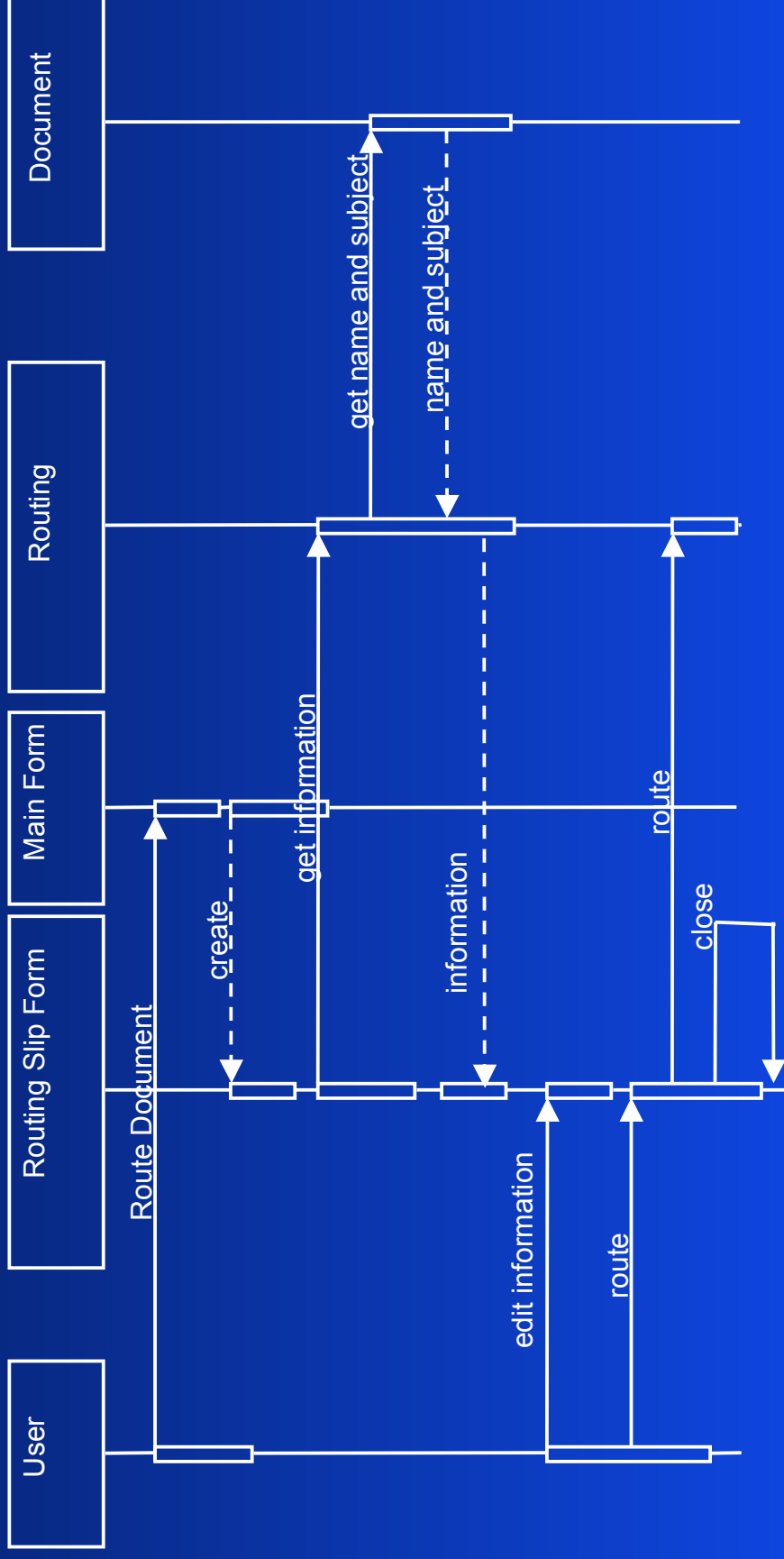
Use case defines menu detail



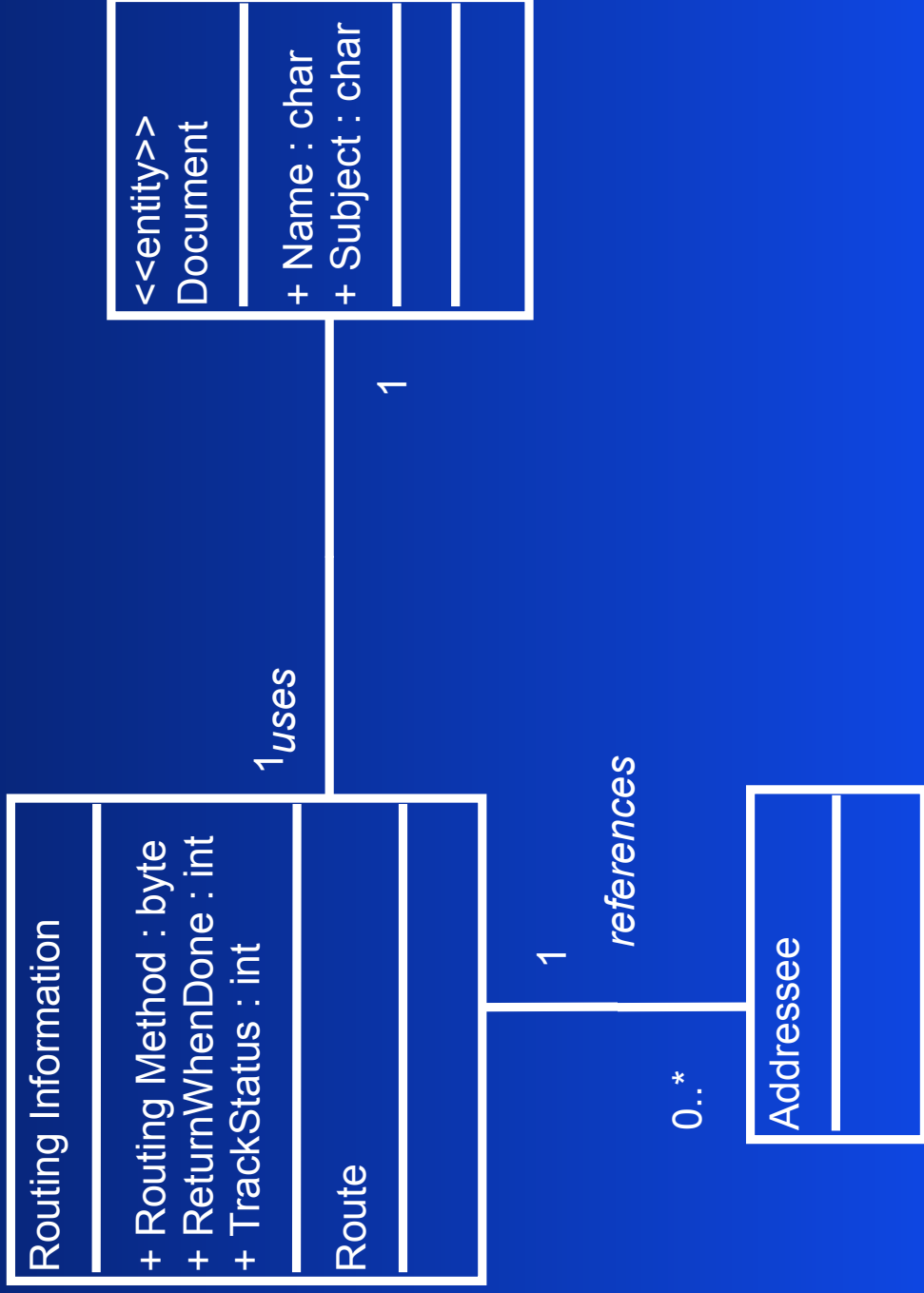
Decomposition ends at atomic processes



Sequence diagrams add temporal information...



Class diagrams provide interaction details for Prototyping



The user interface forms or pages (from previous example) can now be created.

The screenshot shows a 'Routing Slip' dialog box with the following fields and controls:

- From:** Berenbach, Brian
- To:** [Empty text box]
- Subject:** Routing: Converting a UML Model to a Software Prototype
- Message text:** [Empty text box]
- Buttons:** OK, Cancel, Route, Clear, Move, Address..., Remove
- Options:**
 - ☒ Return when done
 - ☒ Track status
 - ☐ One after another
 - ☐ All at once
- Protect for:** Tracked changes

Controls can be inferred from Class diagrams

Item	Displayed As
1:1 Related Object	text box showing name of related object
1:many Related Object(s)	List box or grid double click brings up object
Aggregate 1:1 object	button brings up form showing contained object
Aggregate 1:many objects	List box or grid
Attribute - One of N choices Pull Down	
Attribute – Yes or No	Check Box
Attribute One of N	Radio Button
:	:

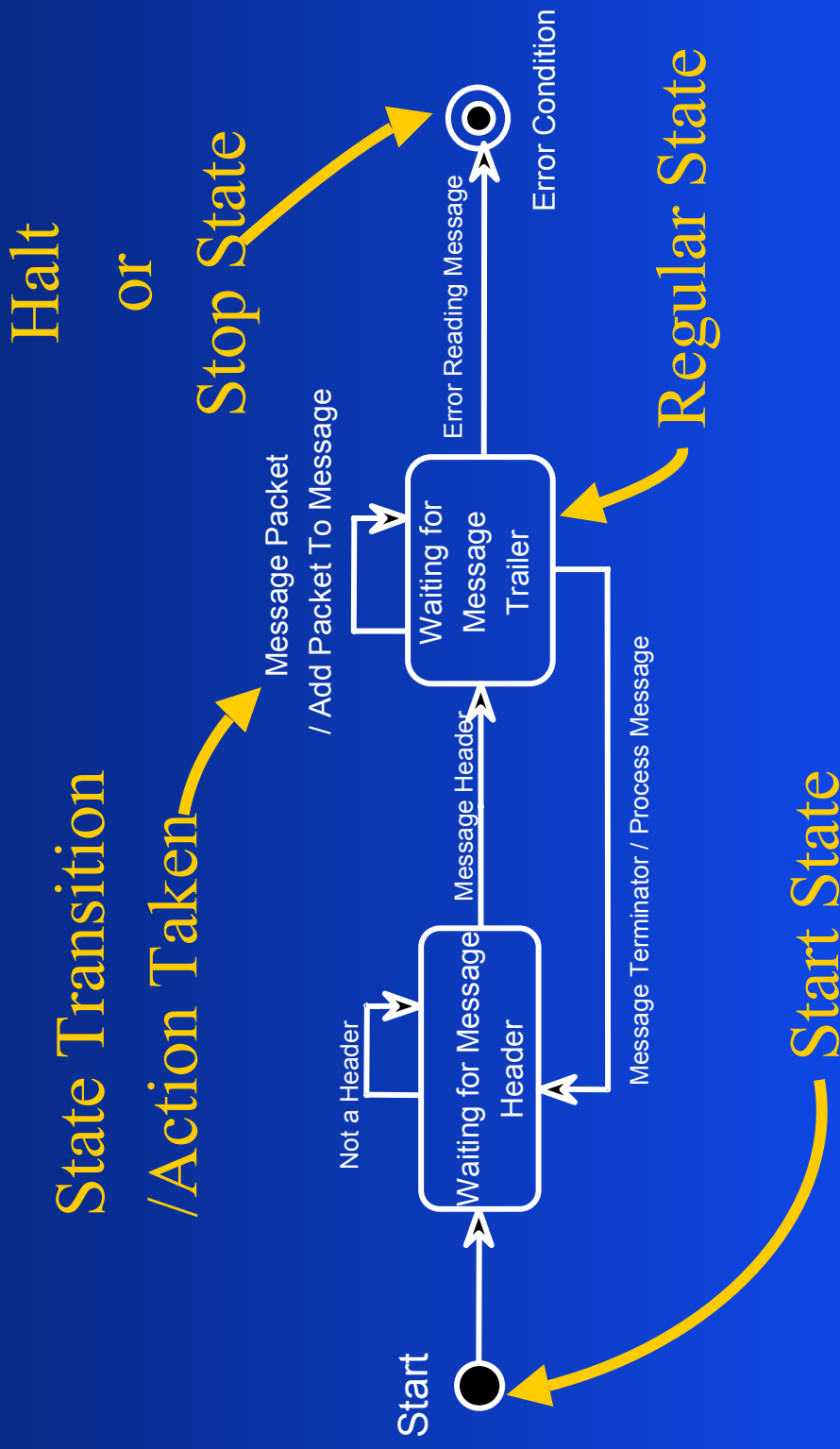
The State Diagram

The State Machine

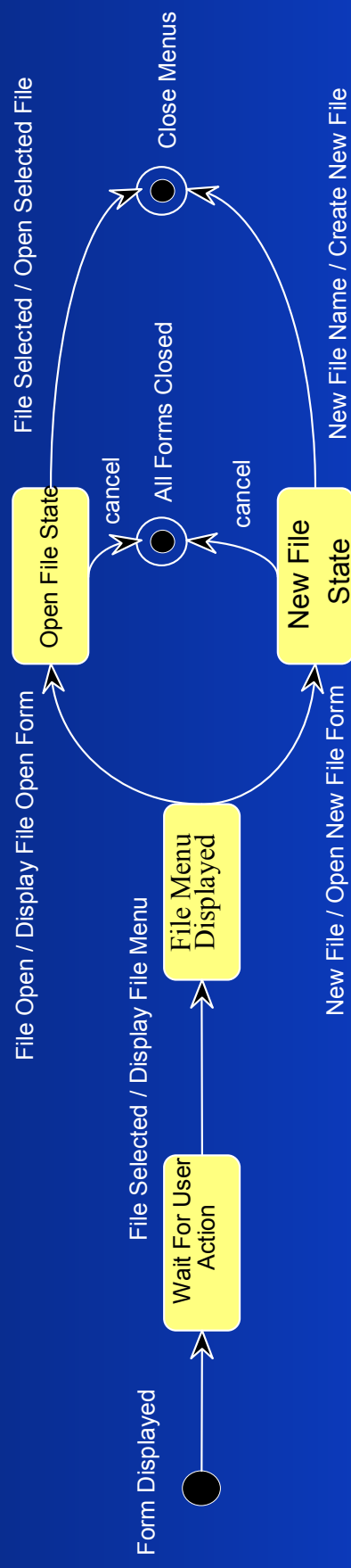
- ❑ Defines the logic for a finite state automaton
- ❑ State changes are driven by inputs
- ❑ The transition is a function only of the current state and the input $T=F(I,S)$.
- ❑ There is a unique start state
- ❑ There may be halt (error) and stop (success) states.

State Diagram Notation

“Process Incoming Messages, where each message will have a single header packet, one or more data packets, and a single trailer packet”



Using State Diagrams to Define Interface Actions



Summary of Steps

- ❑ Create Analysis Standards
 - ❑ Naming Conventions
 - ❑ Use of Diagrams
 - ❑ Organization of Diagrams
 - ❑ Definition of completeness
- ❑ Get all the team members on the same page
 - ❑ Orientation
 - ❑ Training

Summary of Steps

- ❑ Start Modeling from the top
 - ❑ A single entry point
 - ❑ Cross-business
- ❑ Establish Domains
 - ❑ Split into teams (modeler+SME)
 - ❑ Let each team handle one or more domains
- ❑ Rendezvous and Review
 - ❑ Accuracy
 - ❑ Completeness

Sample Correctness Heuristics

- Every use case is defined
- Included and extending use cases are shown on [parent] sequence diagrams
- All actors communicate through interfaces
- All [concrete only!] use cases that communicate with interfaces have the interface on one of their defining sequence diagrams
- There are no cycles in the use case model
- All concrete Classes have instances on Sequence Diagrams
- Everything has an adequate description!!!

Summary of Steps

- ❑ Drive to completion
 - ❑ Every process is adequately defined
 - ❑ All logic exposed
 - ❑ All interfaces exposed
- ❑ Prototype Concurrent with Modeling
 - ❑ Create adequate language/GUI standards
 - ❑ Developers work from the Model
- ❑ Document the completed Model

Summary of Steps

- ❑ Create a tabularized set of requirements
 - ❑ Every major Use case becomes a requirement
- ❑ Add environmental and legacy requirements
- ❑ Prioritize and Cost each Requirement
- ❑ Define the next steps

Questions?