Using UML to Explore Traffic Controller Logic

Prepared By
Larry Head
University of Arizona
Outline

• A Walk through UML
  – Capturing Requirements: Use Cases
  – Structural Modeling:
    • Packages, Objects & Classes
    • Relationships
    • Generalization (Inheritance & Polymorphism)
  – Use Case Realization:
    • Sequence Diagrams
  – Behavioral Modeling
    • Activity Diagrams
    • State Diagrams
  – Implementation & Deployment

• UML for Traffic - Team Modeling Exercise(s)
  – Exercise #1 - Use Case and Class Modeling
  – Exercise #2 - Use Case Realization
  – Exercise #3 - Process (State and Activity) Modeling
  – Exercise #4 - Review the Model (does it work?)
  – Exercise #5 - Implementation and Component

• What’s Next? SysML!
What is UML?

? A new magic way to write software without knowing about C, C++, Java, Ada, Python, ……

? A standardized language from the software industry used to design and specify the functionality of software oriented systems
References
The Unified Process

- Structural View
- Implementation View
- Process View
- Deployment View
- Use Case View
The Unified Process

• Use Case (Requirements) Driven
• Architecture Centric
• Iterative & Incremental
What is a UML Model

Diagrams for Visualization
Context Diagram

Context Diagram is not standard UML diagram type (Class diagram used for this example)
Use Case View
Use Case Specification

- Name/ID
- Actors
- Preconditions
- Primary Flow
  - 1. The use case begins
- Post-conditions
- Alternative Flow
- Derived Requirements

<table>
<thead>
<tr>
<th>Use case</th>
<th>BrowseProducts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use case ID</td>
<td>UC8</td>
</tr>
<tr>
<td>Actors</td>
<td>Customer</td>
</tr>
</tbody>
</table>

**Preconditions**
None

**Flow of events**
1. The use case begins when the Customer selects a type of product to browse.
2. The system displays a list of all categories for that type of product.
3. The Customer selects a category.
4. The system displays a page containing a maximum of 10 products in the category. This page includes summary information for each product and its price.
5. While the Customer is browsing
   5.1 If there are more products to display
      5.1.1 The Customer may select “Next” to view the next page of products.
      5.2 If the Customer is not on the first page of products
      5.2.1 The Customer may select “Previous” to view the previous page of products.
6. The Customer selects a product.
7. The system displays full details of the product including its price.
   <additem>

**Postconditions**
1. The Customer has selected a product.
2. The system has displayed full details of the product.

**Alternative flow 1**
1. At any point, include( FindProduct ).
Design View

Segment
1. The system retrieves the Customer’s credit card details.
2. The system sends a message to the CardProcessingCompany with identifier, merchant authentication, name on card, number of card, amount of transaction.
3. The CardProcessingCompany authorises the transaction.
4. The system notifies the Customer that the card transaction has been completed.
5. The system gives the Customer an order reference number for the transaction.
6. The system tells the Inventory agent to reserve the required items.
7. The system sends the order to the Dispatcher.
8. The system changes the order’s status to pending.

Class Diagram

Class Domain Model
- "entity" Customer
  - id: int
  - name: string
- "entity" CreditCard
  - expiryDate: int
  - nameOnCard: string
  - number: int
  - type: enum
- "entity" shoppingCart
  - orderReferenceNumber: int
  - total: float
- "entity" Message
  - transactionMessage
    - merchantInfo
      - authentication: int
      - id: int
- "boundary" CardProcessingCompany
  - send(transactionMessage): boolean
- "boundary" inventoryManager
  - products
- "boundary" transactionControl
  - products
What is a Class?

• A “noun” in the use cases
• Has clear role and responsibility
• Data+Behavior
Design View - Stereotypes

- **<<entity>>**: A class that contains persistent data
- **<<control>>**: A class that contains the details of an algorithm
- **<<boundary>>**: A class that knows how to interact with an outside system
Design View - Objects: Instances of Classes

Class

- id: int
- name: string

Objects

«entity»
JohnSmith:Customer

«entity»
CityofDetroit:Customer

«entity»
OaklandCounty:Customer
Design View - Packages

- Container for organization
- Separate functions and work so Teams can do development
- Namespace - variables
Design View - Relationships

- **Association**
- **Dependency**
- **Aggregation**
- **Composition**
- **Generalization**
- **Realization**
- **Interface**
Process View - Use Case Realization: Sequence Diagrams

Objects

Lifeline

Operations “Messages”

Time

?? Catalog -> CardProcessingCo
Design View - Class Diagram (Original)
Revised Class Diagram
Revised Sequence Diagram
Process View - Activity Diagram
Process View - State Diagram

- Class Order
  - date: int
  - number: int
Implementation View - Components

- Software Components that can be Deployed on devices that have processors & memory (called Nodes)
Implementation View - Components
Implementation View - Components
Implementation View - Components

```
<control>
Domain Model::transactionManager
+ checkCredit(): boolean
+ processCreditCard(): boolean
+ processProductOrder(): boolean
</control>

<entity>
Domain Model::shoppingCart
- orderReferenceNumber: int
- total: float
+ addItem(products): void
+ checkOut(): void
+ removeItem(products): void
+ total(): void
</entity>

<entity>
Domain Model::CreditCard
- expDate: int
- nameOnCard: string
- number: int
- type: enum
</entity>

<entity>
Domain Model::Customer
- id: int
- name: string
</entity>

<entity>
Domain Model::Order
- date: int
- number: int
</entity>
```
Deployment View
Exercise #1: Use Case + Classes

- Consider basic traffic control
  - Group: Brainstorm Use Cases
  - Breakout:
    - Use Case Specs
    - Classes
    - 20 Minutes
Use Cases

• Use Cases
  – Detect Cars
  – Control Flow on Movement
  – Detect Peds
  – Detect EV’s
  – Detect Train
  – Detect Transit Vehicles
  – Report Status/Faults
  – Maintain Coordination
  – Modify Operational Parameters/Config
  – Perform Diagnostics
  – Log events
  – Collect Performance Data
  – Manual Control
  – Priority Treatment

• Actors
  – Vehicle
  – Ped
  – EV
  – Transit
  – RailRoad
  – Time
  – Central System
  – Technician
  – On-Street Master
  – Police
Exercise #2: Use Case Realization

• Given Use Cases and Class Definitions
  – Generate a sequence diagram for your Use Case
  – Add Operations and Attributes as needed
  – 20 Minutes
Exercise #3: Process (State and Activity) Modeling

- Define the logic to implement the Behavior in the Sequence Diagrams
  - Generate an Activity Diagram for one Behavior
  - Generate a State Model for one Entity/Activity
  - 20 Minutes
Exercise #4: Review the Model - Does it work?
Exercise #5: Implementation & Component

• Define software components
• Define hardware and deployment components
  – Alternatives?
Summary Comments

• UML is a language to describe software
• The Unified Process
  – Use Case (Requirements) Driven
  – Architecture Centric
  – Iterative & Incremental
• What’s Next? SysML
  – www.omgsysml.org
SysML Diagram Taxonomy

- Activity Diagram
- Sequence Diagram
- State Machine Diagram
- Use Case Diagram
- Block Definition Diagram
- Internal Block Diagram
- Package Diagram
- Requirement Diagram
- Structure Diagram
- Behavior Diagram

- Same as UML 2
- Modified from UML 2
- New diagram type

Relationship Between SysML and UML

- UML reused by SysML (UML4SysML)
- UML not required by SysML (UML - UML4SysML)
- SysML extensions to UML (SysML Profile)

SysML Extensions:
- Blocks
- Item flows
- Value properties
- Allocations
- Requirements
- Parametrics
- Continuous flows

Block Definition vs. Usage

**Block Definition Diagram**

```
bdd [package] VehicleStructure [ABS-Block Definition Diagram]
```

- «block» Library:: Electronic Processor
- «block» Anti-Lock Controller
- «block» Traction Detector
- «block» Brake Modulator
- «block» Sensor

**Internal Block Diagram**

```
ibd [block] Anti-LockController [Internal Block Diagram]
```

- c1:
  - s1:Sensor
- c2:
  - d1:Traction Detector
  - m1:Brake Modulator

**Definition**
- Block is a definition/type
- Captures properties, etc.
- Reused in multiple contexts

**Usage**
- Part is the usage in a particular context
- Typed by a block
- Also known as a role
Defining Vehicle Dynamics

```
package Analysis

parametric Diagram

constraint StraightLineVehicleDynamics
    V: VehicleStructure

constraint BrakingForceEquation
    constraints: f = (f*bf)*(1-tl)

constraint AccelerationEquation
    constraints: F = m*a
    parameters: F: force, m: mass, a: acceleration

constraint VelocityEquation
    constraints: a = dv/dt
    parameters: a: acceleration, v: velocity, t: time

constraint DistanceEquation
    constraints: v = dx/dt
    parameters: v: velocity, x: position, t: time
```

Defining Reusable Equations for Parametrics

Vehicle Dynamics Analysis

par [constraintBlock] StraightLineVehicleDynamics [Parametric Diagram]

- v.chassis.tire.friction:
  - tf:
  - tl:
  - bf:

- v.brake.abs.m1.duty_cycle:

- v.brake.rotor.braking_force:
  - f:

- v.mass:
  - m:

«constraint»
e1:BrakingForceEquation
[f = (tf*bf)*(1-tl)]

«constraint»
e2:AccelerationEquation
[F = m*a]

«constraint»
e3:VelocityEquation
[a = dv/dt]

«constraint»
e4:DistanceEquation
[v = dx/dt]

Using the Equations in a Parametric Diagram to Constrain Value Properties