Domain Models and UML
Object Oriented Analysis and Design

Aron Trauring
T++ Technical Skills Training Program
CUNY Institute for Software Design & Development (CISDD)
New York Software Industry Association (NYSIA)

December 8th, 2004
Requirements Specification When?

- SDLC — Analysis Phase
- UP — Requirements Workshop
- XP — Release Planning Game and throughout the process
Requirements Specification Artifacts

• Use Cases/Stories

• Supplementary Specification — non-functional requirements

• Glossary

• Domain Rules — how the Domain (organization) operates — application independent
OO Analysis Artifacts

• Domain Models

• System Sequence Diagrams (SSD)

• Operation Contracts
Domain Model Defined

- OOA — Decomposition of a domain into noteworthy concepts or objects

- DM — visual representation of above in simple UML

- Other names: conceptual models, domain object model, analysis object model

- Bounded by a specific domain as described in use cases

- Concepts not software classes (these are in domain layer)
Domain Models — What?

- “Visual dictionary of noteworthy abstractions, domain vocabulary and information content”

- Domain concepts — noteworthy abstractions, domain vocabulary — idea, thing or object (Payment, Sale)

- Associations — relationships between concepts (Payment Pays-For Sales)

- Attributes — information content (Sale records date and time)
Domain Models — What Not?

- Think about the domain not the software

- Software artifacts (SalesDatabase)

- Object responsibilities or methods (e.g. Sale has a print() method)

- Not a Data Model — Do include concepts that have no attributes or play a behavioral role (Customer)
Domain Concepts (Conceptual Classes)

- Symbol — words or images representing the concept
- Intention — definition of the concept
- Extension — the set of examples to which the concept applies
Example — Purchase Transaction
"A sale represents the event of a purchase transaction. It has a date and time."

Sale

concept's symbol

date
time

concept's intension

concept's extension

sale-1

sale-2

sale-3

sale-4

Aron Trauring — Zoteca
Associations

• Meaningful and Interesting relationship or connection
Attributes

• Logical data value of a concept
Domain Models — Why?

- Domain Model is mirrored in Domain Layer — low representational gap

- Focus on customer needs — customer domain becomes language of developers
UP Domain Model

Stakeholder's view of the noteworthy concepts in the domain.

conceptual classes

<table>
<thead>
<tr>
<th>Payment</th>
<th>1 Pays-for 1</th>
<th>Sale</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount</td>
<td></td>
<td>date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>time</td>
</tr>
</tbody>
</table>

inspires objects and names in

design classes

<table>
<thead>
<tr>
<th>Payment</th>
<th>1 Pays-for 1</th>
<th>Sale</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount: Money</td>
<td></td>
<td>date: Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>startTime: Time</td>
</tr>
<tr>
<td>getBalance(): Money</td>
<td></td>
<td>getTotal(): Money</td>
</tr>
</tbody>
</table>

UP Design Model

The object developer has taken inspiration from the real-world domain in creating software classes. Therefore, the representational gap between how stakeholders conceive the domain, and its representation in software, has been lowered.
How to Create a Domain Model — Three Steps

1. Find the domain concepts

2. Draw them in a UML class diagram

3. Add associations and attributes
Three Strategies for Finding Domain Concepts

1. Reuse Existing Models

2. Category Lists

3. Linguistic Analysis
Reuse Existing Models

- Best strategy
- Past models
- “Analysis patterns”
Category Lists

- Larman p. 140-141
- Focus on *noteworthy* concepts
- Linguistic Analysis
Noun phrase identification

- Extremely easy
- Mechanical application leads to trivial and/or mistaken models
- Use with category lists
- Sources: Use cases, domain experts, other documents
- Customer arrives at POS Checkout with goods to purchase
- Ambiguities: goods, products, items
Narrowing the List

• Limit model to domain of iteration (look at chosen use cases)

• Make list of “candidates” and then think — focus on noteworthy

• Use customer terms — SKU or Item
Sketches vs. Tools

- Work on whiteboards and take digital photos
- Only save in tool if domain will revisit often
- Don't spend much time reworking and updating models
Description Classes (Larman)

- Flight vs. FlightDescription

- Larman's motivation is implementation specific

- FlightDescription is perhaps more an attribute
Tools — Drawing Programs

- Open Office http://www.openoffice.org/
- Dia http://www.gnome.org/projects/dia/
- Violet http://www.horstmann.com/violet/
- Visio
Tools — Simple UML Modelers

- ArgoUML http://argouml.tigris.org/
- Umbrello http://uml.sourceforge.net/index.php
Tools — Eclipse Plugins with Free Versions

- Omondo http://www.omondo.com/index.html
- Visual Paradigm http://www.visual-paradigm.com/
Tools — Commercial with Community Editions

- Poseiden http://www.gentleware.com/
Tools — Commercial

- Rational (IBM)

    The traditional Rational message has been that we are agnostic to the particular languages and platforms used to implement solutions. That was yesterday. IBM is designing and delivering the industry’s leading computing platform for the implementation, delivery, and management of enterprise solutions. The new Rational role is to encourage, support, and help differentiate that platform from our competitors.


- Together (Borland)
• Microsoft will “extend” UML in Longhorn
Mini Exercise 1 — Entering Classes
Associations — When?

- Limit number to avoid diagram clutter
- Not meant to document software object or data structures
- Relationship that needs to be preserved for some duration
- Need to preserve memory of relationship between the concepts
- Real world need not implementation need
- Sales recorded in a Ledger --- completed Sales need to be remembered
- TicketAgent looks up FlightInformation — not necessary to remember once flight is booked
Associations — How?

- Common Associations List Larman p. 155-156)

- Verb phrases in use cases
Association Notation

- "reading direction arrow"
- it has no meaning except to indicate direction of reading the association label
- often excluded

<table>
<thead>
<tr>
<th>Register</th>
<th>1</th>
<th>Records-current</th>
</tr>
</thead>
<tbody>
<tr>
<td>association name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Sale | 1 |
| multiplicity |
Association Name Conventions

- Capitalized (Paid-by or PaidBy)
- Meaningful Verb-Phrase
- Enhance understanding (Paid-by — Good, Uses — Bad)
Association-Ends — Roles (Optional)

• Name

• Multiplicity Expression

• Navigability
Association Role Example

- Role name alternative to association name
- Navigability — uni- or bi-directional
- Multiple associations between two concepts possible (Flight <-> Airport)
Multiplicity — Notation

- **1** — An order must have exactly one customer

- **0..1** — A corporate customer may or may not have a single sales rep

- ***** — A customer need not place an Order and there is no upper limit to the number of Orders a Customer may place (zero or more orders)
Multiplicity — Meaning

- Domain constraint of importance
- Usually related to implementation checking
- Multiplicity is context dependent

Person Works-for 1 Company (payroll system)

Person Works-for * Company (tax system)
Mini Exercise 2 — Entering Associations
Attributes — When?

- Requirements suggest or imply a need to remember information

  Sale needs a dateTime

  Store needs a name and address
Attributes vs. Domain Concepts vs. Associations

- Something which is a number or text (primitive data type) in real world is most likely an attribute

  Phonenumber, Name, Date

- Complex domain concepts should be related by associations not attributes
• Don’t show complex concepts as attributes; use associations
• Use associations instead of attributes as foreign keys (design creep)
Attribute — Notation

• Shown in second compartment

• visibility derived name: type multiplicity = default {property-string
  
  - name: String [1] = "Untitled" {readOnly}

• Besides name, all other information optional

• Visibility — indicates whether the attribute is public (+) or private (-)

• Derived — / indicates derived from other information — / total : Money

• Type — restriction on what kind of object may be placed in the attribute.
• `{property-string}` allows you to indicate additional properties for the attribute
Attribute — Type

• In general primitive data type — Boolean, Number, String

• Units, quantities, time should be modeled
• Complex data types should be defined — guidelines on Larman p.164

• Larman says model these — put in glossary instead
Mini Exercise 3 — Entering Attributes