

Object Modeling with UMLTM

Steve Tockey
Construx Software Builders
14715 Bel-Red Rd #100
Bellevue, WA 98007

stevet@construx.com

www.construx.com



Section Outline

- Goals of This Tutorial
- Outline of This Seminar
- Why Have Software Models, Anyway?
- Understanding Software Methods
- The History of UML
- Generic UML Facilities
- For More UML Information...

Goals of This Tutorial

- To have you understand a practical and useful distinction between “analysis” and “design”
- Give you a tour of a subset of UML in the context of this definition of “analysis” and “design”
- To give you a set of references so that you can get more information about UML after this tutorial is over

Outline of This Tutorial

- Analysis vs. Design: What's the Difference?

- UML Meets Analysis
 - Use Case Diagrams
 - Class Diagrams
 - Collaboration Diagrams
 - State Diagrams, Actions, and Activities

- Moving into Design

- Tutorial Summary

Why Have Software Models, Anyway?

- A study by the US General Accounting Office showed a 4% probability of success in delivering software, at cost and on schedule, that actually met customer needs [complete reference unknown]
- A 1975 study showed that a majority of software errors in delivered systems are not coding related
 - Instead, they are a direct result of missing, conflicting, mis-understood, or mis-interpreted requirements [Boehm75]

In other words, the code did exactly what the programmer wanted. The trouble was, customer wanted something else

[Boehm75]

Barry W. Boehm, *Software Design and Structuring*, in Practical Strategies for Developing Large Software Systems, Ellis Horowitz ed., Addison-Wesley, 1975

Why Have Software Models, Anyway? (cont)

- The focus of software methods is on “blueprints” that help us build it right the first time

"The overriding concern ... is not to achieve success, but to avoid failure." [DeMarco79]

- Industry studies also show that about 80% of the total software lifecycle cost is maintenance (i.e., post-delivery)
 - Properly maintained, these same blueprints can significantly reduce software maintenance costs by answering the maintainer's key questions:
 - * What is this piece of the software for?
 - * Why does it look the way it does?

[DeMarco79]

Tom DeMarco, Structured Analysis and System Specification, Yourdon Press, 1979 (p9)

Understanding Software Methods

- In general, modeling methods can be described in four separate dimensions
 - Concepts
 - Rules
 - Notations
 - Process(es)

- Concepts
 - The aspects of a system that can be captured in that kind of model

What kinds of things about the world can be modelled?

- Rules
 - The criteria for determining whether or not a given model is well-formed

How do you know that a model is meaningful?

- Notations
 - The graphical and textual forms for rendering models

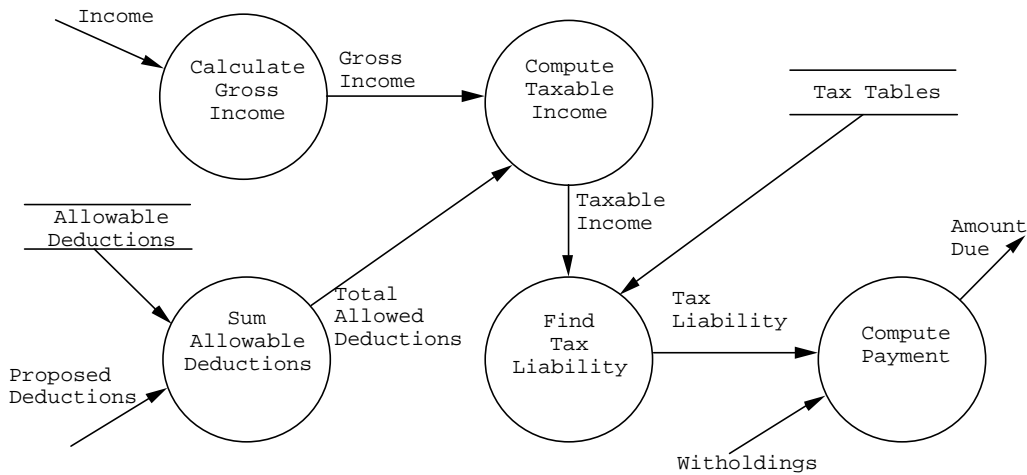
How do you write the model down so that others can read it?

- Process(es)

- The ordered steps that lead you to completion
How do you get from a “blank sheet of paper” to a finished model?

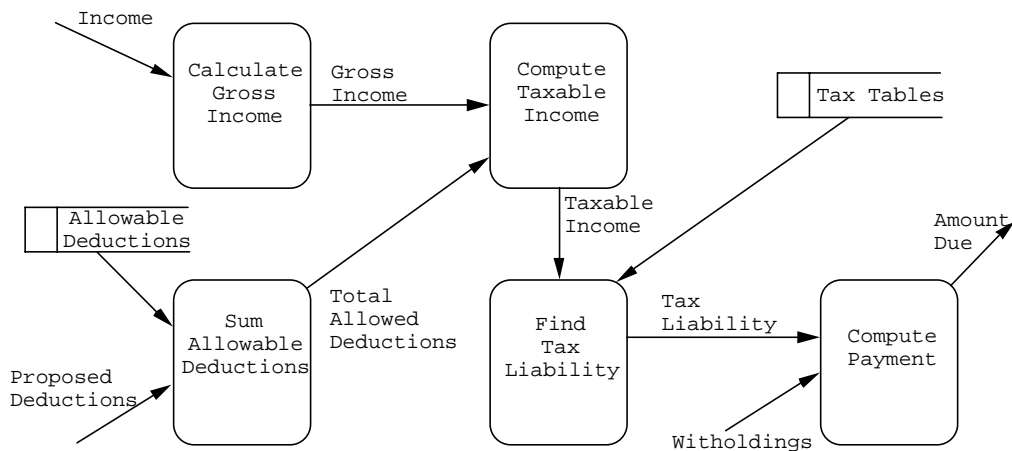
Understanding Software Methods (cont)

- What are the concepts and rules in this example?



- What are the notations in the example? What are the process(es)

- What about the following model?



Understanding Software Methods (more)

- Why is any of this important?
 - The concepts and rules frame your ability to model
 - Notations give you a means to share the model with others
 - * But the same concepts can often be expressed with different notations
 - Different process(es) work better in different situations
 - * There is no one-size-fits-all process

- So what does this have to do with UML?
 - UML has a defined set of concepts
 - UML has a defined set of rules
 - UML has a defined notation
 - UML does not have any defined process(es)

The History of UML

- Object-oriented design first appeared in about 1984

- Object-oriented analysis first appeared in about 1987

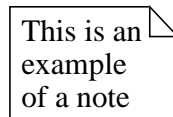
- Since then, a number of “competing” approaches have emerged (Booch, Rumbaugh, Shlaer-Mellor, ...)
 - Most of these approaches share many common features
 - Most of these approaches have arbitrary differences

- The Object Management Group (OMG) started an effort in June, 1995 to agree on a common approach to modeling object-oriented systems
 - UML 1.1, formally adopted by the OMG on November 19, 1997
 - UML 1.2, editorial revision
 - UML 1.3, minor updates based on user comment
 - UML 1.4, TBD
 - UML 2.0, substantial internal restructuring

Generic UML Facilities

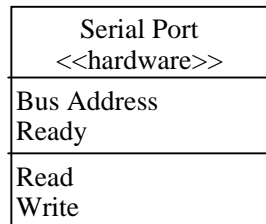
- Notes

- A graphical symbol containing textual information (possibly including embedded images)
 - * UML's "comment"
 - * Can be "attached" to a model element by a dashed line



- Stereotypes

- A way to specify a usage distinction
- Can either use <<stereotype name>> or a special icon



- Types vs. instances

- Use the same icon, but underline the name

Bank Account
Balance Status
Open Deposit Examine Balance Withdraw Close

Account 000295:
Bank Account

Generic UML Facilities (cont)

- Constraints

- Semantic relationships among model elements that specify conditions and propositions that must be maintained as true
- No defined UML constraint language, although OCL is part of the UML definition

```
{ Employee.Salary < Employee.Boss.Salary }
```

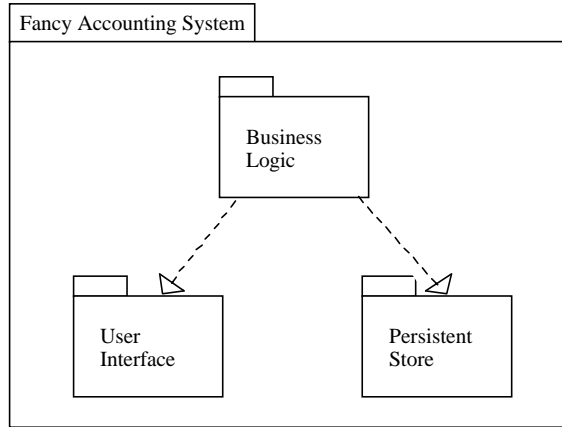
- Tagged Values

- General properties attached to model elements

```
{ author = Joe, status = tested }
```

- Packages

- A way to break models into more manageable chunks
- The dashed-line-with-arrow means a dependency exists between the packages
- Use `PackageName::ElementName` to refer to a model element in a different package



For More UML Information...

- http://www.omg.org/technology/documents/formal/unified_modeling_language.htm
- Craig Larman, *Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design*, Prentice-Hall, 1997
- Martin Fowler with Kendall Scott, *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley, 1997
- Hans-Erik Eriksson and Magnus Penker, *UML Toolkit*, Wiley, 1997
- James Odell and Martin Fowler, *Advanced Object-oriented Analysis and Design using UML*, SIGS Books, 1998
- Pierre-Alain Muller, *Instant UML*, Wrox Press, 1997
- James Martin and Jim Odell, *Object-oriented Methods: A Foundation: UML Edition*, Prentice-Hall, 1997
- Bruce Powell Douglass, *Real-time UML: Developing Efficient Objects for Embedded Systems*, Addison-Wesley, 1998
- Paul Harmon and Mark Watson, *Understanding UML: The Developer's Guide: With a Web-based Application in Java*, Morgan Kaufman, 1997
- Putnam Texel and Charles Williams, *Use Cases Combined with Booch/OMT/UML: Process and Products*, Prentice-Hall, 1997
- Magnus Penker and Hans-Erik Eriksson, *Business Modeling with UML: Business Patterns and Business Objects*, Wiley, 1999

For More UML Information... (cont)

- R J Pooley and Perdita Stevens, Component Based Software Engineering with UML, Addison Wesley, 1998
- Jos Warmer and Anneke Kleppe, The Object Constraint Language: Precise Modeling with UML, Addison-Wesley, 1998
- Desmond D'Souza and Alan Wills, Objects, Components and Frameworks with UML: The Catalysis Approach, Addison-Wesley, 1998
- Ari Jaaksi et al, Tried and True Object Development: Industry-proven Approaches with UML, SIGS Books, 1998
- Sinan Si Alhir, UML In a Nutshell, O'Reilly & Associates, 1998
- <http://www.rational.com>
- OTUG@rational.com
- ADTF@omg.org
- UML-RTF@omg.org

Key Points

- Methods result in “blueprints” for software systems
- These “blueprints” are extremely valuable during maintenance
 - Where about 80% of the lifecycle cost really is
- In general, modeling methods can be described in four separate dimensions
 - Concepts
 - Rules
 - Notations
 - Process(es)
- UML only defines Concepts, Rules, and Notations
 - Process is not part of the existing UML definition
- UML is a joint effort sponsored by the OMG
- There are a number of generic UML facilities
 - Notes
 - Stereotypes
 - Types vs. Instances
 - Constraints
 - Packages
- There are plenty of resources for finding more UML information