Agenda

• What is UML 2?
• UML 2 Overview – work through the diagrams
• Extending UML
• MDA
What is UML 2?
Model Driven Development

- Traditionally, engineers model before they build the real thing.
- Model in abstractions that are easier to understand than the real thing.
- Carefully chosen abstractions/models promote:
  - Communication
  - Validation
- Before going to the expense of full construction.
UML – a Visual Modelling Language

- **UML – Unified Modelling Language**
  - Is a visual modelling language
  - Describes the specification and design of a software intensive system via visual models

Models
- Realised-by
  - Business Analysis Model
- Implemented-by
  - Design Model
- Verified-by
  - Test Model

- Realised-by
  - Business Use Case Model
- Implemented-by
  - Use Case Model
- Verified-by
  - Automated-by
  - Realised-by
  - Test Model
UML 2 – what is its pedigree?

- **Industrialisation**
  - UML 2.0 - MDA
  - Q1 03
  - Oct 04 – final OMG endorsement

- **Standardisation**
  - UML 1.5
  - 2001
  - UML 1.4 (action semantics)
  - 1998
  - UML 1.3 (extensibility)
  - 1997
  - UML 1.1 (OMG Standard)
  - 1996

- **Unification**
  - Booch,
  - Rumbaugh,
  - Jacobson,
  - 92-96

- **Fragmentation**
  - The method wars
  - Foundations of OO – Stroustrup, Harel, Wirfs-Brock, Meyer etc,
An Architectural Representation

- Conceptual View
- Process View
- Logical View
- Use-Case View
- Implementation View
- Deployment View

End-user
Functionality

Analysts/Designers
Structure

Programmers
Software management

Designers / Integrators
Performance
Scalability
Throughput

System engineering
System topology
Delivery, installation
communication

Conceptual

Physical
Overview of UML2
Whats New Summary

Diagram

Structural
- Class
- Object
- Package
- Deployment

Behavioural
- Activity
- State
- Usecase
- Interaction
- Protocol State

Component
- Composite Structure

Unchanged

Changed

Significant Change

New

Sequence

Timing

Communication
Use Case View – Use Case Diagram

- Show all the ways of using the system
- Diagram is 20% of the effort
  - 80% is writing textual use case specifications

Use-Case Specification

<table>
<thead>
<tr>
<th>Name</th>
<th>Brief description</th>
<th>Flows of events</th>
<th>Special requirements</th>
<th>Preconditions</th>
<th>Postconditions</th>
</tr>
</thead>
</table>

Nursery Automation System
Use Case Diagram - 2

- Still has
  - Include, extend and generalisation
- Organise large use case model
  - As set of diagrams organised by function or by actor
• Classifiers (class, interface, signal node etc) can now own use cases

• When would you use this?
  – Model an enterprise architecture as a set of collaborating components
  – Each component could then ‘own’ its own requirements

• Associated Diagrams
  – Activity Diagram – can be used to model interaction between scenarios
  – Interaction and Class diagrams model the use case realisation
Activity Diagram

• **Focus on**
  – Flow of activities involved in a process
  – NB: State Diagram focuses on an object undergoing a process

• **Significant Changes from UML 1.x**
  – Now address real time flow, incorporated Petri Net concepts (tokens)
  – Decompose activity into multiple actions
  – Activity parameters and pins

• **Useful because they focus on workflow**
  – Don’t have to reference a particular object so ideal to map complex alternative flows in a use case

• **Visual map of a set of activities and possible transitions between them**
  – Activities typically fulfilled by one or more operations
Activity Diagram - 1

Activity Partition (swim-lane)

Start

Insert Card

Enter Pin

Enter Amount

Take Money from Slot

Take Card

ATM

Prompt Pin

Prompt Amount

Output Cash

Eject Card

Display Balance

Debit Account

Amount

Bank Back Office

Authorise

Check Balance

Valid Pin

[Invalid Pin]

[Balance ≤ Amount]

Stop

Fork

Guard

Join

Object – can act as a Data-store retaining a value between actions

Action

Branch
Activity Diagram - 2

- Signals are modelled as:
  - Send Signal Action
  - Accept Event Action

- Repetitive Time Events
  - Except Time Event Action
Activity Diagram – 3 Activity Parameters and Pins

Activity Parameters

Activity

Input Pin

Output Pin

Flow Final – consumes a token Without terminating the activity

Append Post Codes

Address

getAddress

Append Post Code

Update Address

Complete Address

Post Code
Activity Diagram – 4 Advanced

Indicates Concurrency

Expansion Region

Process Personal Application

Verify Completeness

<<decisionInput>> Eligibility Rules

Process Ineligible Application [Person Not Eligible]

Check Eligibility [Person Eligible]

Valid Application

Structured Activity

Send Confirm Notice

Post Confirmation

Store Valid Application

<<Structured>>

Information collection

Valid Application

IO Exception

Handle IO Exception

SQL Exception

Handle SQL Exception

Multiple Exception Handlers – normally Keep at abstraction level Of figure

NB – implementers ensure Pre/post conditions met, Manage access to shared Resources in structured activity
What's New Summary

Diagram

- Structural
  - Class
  - Object
  - Package
  - Deployment
  - Component
  - Composite Structure

- Behavioural
  - Activity
  - State
  - Use case
  - Interaction
  - Protocol State

Unchanged

Changed

Significant Change

New
Logical View - Class Diagram

• **Shows static structure**
  – The ‘things’ - classes that are handled in the system
  – Relationships between classes
    • Associated (connected)
    • Dependent (uses)
    • Specialised
    • Packaged – grouped together as a unit

• **It’s a static view**
  – Structure described always valid irrespective of where the system is in its lifecycle

• **Typically**
  – Draw multiple class diagrams
  – Classes can appear on multiple diagrams
  – Not all classes appear on a single diagram
Class Diagram - Generalisation

- Gender
- Employment Status
- Sets
  - {incomplete, overlapping}
    - Propulsion
  - {incomplete, disjoint}
    - Licensed

Generalisation Sets – attributes – constrains the generalisation set

PowerType
Interfaces

- Introduced Ball-Socket notation
- Remains a ‘contract’
- Really comes into its own on the component diagram
  - Component now plays the role of the sub-system used too
What's New Summary

Diagram
- Structural
  - Class
  - Object
  - Package
  - Deployment
  - Component
    - Composite Structure
  - Interaction Overview
- Behavioural
  - Activity
  - State
  - Use Case
    - Protocol State
  - Interaction
    - Sequence
    - Timing
    - Communication

Unchanged
- Changed
- Significant Change
- New
Package Diagrams

- Organises the model
- Dependency indicates a usage relationship between contents of the packages
- Use it to model the high level organisation of the system
- Indicate visibility on owned elements
  - Indicates their availability from other packages
- Import from packages
  - <<import>> public
  - <<access>> private
- Supports Package Merge
  - Use to merge elements of same name in different packages into one
Ports (New)

• Ports are a big change
• Introduced from ROOM
• Defines an interaction point on a classifier
  – Between it and its environment
  – Interaction is formalised
    • externally via either a supplied or required interface
    • Internally via a protocol state machine

Formalises – design by contract
What's New Summary

Diagram

Structural
- Class
- Object
- Package
- Deployment

Behavioural
- Activity
- State
- Use Case
- Interaction
- Protocol State

Component
- Composite Structure

Unchanged
- Changed
- Significant Change
- New
Composite Structure Diagram (New)

- UML 1.x only had composition to model internal structure
- UML2 introduces this diagram to model ‘hidden’ detail

- Models what happens on instantiation of a classifier
  - Instantiation of a car or a boat results in the internal structures below
  - Emphasises the principles of composition
    - Car owns all contained instances

- A dotted outline indicates a weaker reference relationship
What's New Summary

Diagram

Structural
- Class
- Object
- Package
- Deployment
- Component
- Composite Structure

Behavioural
- Activity
- State
- Use Case

Interaction
- Interaction Overview
- Sequence
- Timing
- Communication

Unchanged

Changed

Significant Change

New

Protocol State
Protocol State Machine (New)

- Shows the lifecycle of an instance as it responds to external events
  - The protocol specifies the rules for how the classifier can execute behaviour
  - Usually relates to a specific port/interface
  - Detailed internal behaviour defined on the behavioural state machine
    - To comply with the protocol rules

- Associated with a port, defines
  - Rules to be followed by all instantiations
  - All legal transitions for each operation
Behavioural State Diagram

- Defines classifier life-cycle
  - Define the states
  - Legal transitions between the states in receipt to events
    - Received messages
    - Time elapsed
    - Errors
    - Conditions becoming true

- Use when a classifier has significant state dependent behaviour
**Behavioural State Diagram - 2**

- **Nesting Diagrams**
  - Manages complexity
  - Orthogonal sub-states (or sub-states)
The running state can have concurrent sub-states

- Non-orthogonal (and sub-states)
- Also known as concurrent sub-states
  - Use when modelling states of concurrent threads
Sequence Diagram (Major Changes)

- Now have a frame
- Show mini sequence diagrams within main diagram (Fragments)
  - ref (reference)
  - alt (alternative)
  - Loop
  - par (parallel)
  - crit (critical region)
  - strict (strict sequencing)
  - seq (weak sequencing)

- Can be used in
  - Instance form
    - Models a particular scenario in detail
  - Generic form
    - Maps all scenarios

![Sequence Diagram](image)
Whats New Summary

Diagram

Structural
- Class
- Object
- Package
- Deployment

Behavioural
- Activity
- State
- Usecase
- Interaction
- Protocol State

Component
- Composite Structure

Interaction Overview

Sequence
Timing
Communication
Interaction Overview Diagram (New)

- Describes the interaction between the possible many flows of a complex use case or system
  - Variant of Activity Diagram
- Where to use it?
  - Model how the different scenarios of a use case are realised together
  - Model an interesting set of interactions across the system
What's New Summary

Diagram

Structural
- Class
- Object
- Package
- Deployment

Behavioural
- Activity
- State
- Use case
- Interaction
- Protocol State

Component
- Composite Structure

Unchanged
- Changed
- Significant Change
- New
• Equivalent of UML 1 Collaboration Diagram
• Shows the network of objects to carry out a goal
  – Can’t show structure (alt, ref etc) from sequence diagrams
  – Don’t show time-line
What's New Summary

Diagram

Structural
- Class
- Object
- Package
- Deployment
- Component
- Composite Structure

Behavioural
- Activity
- State
- Interaction
- Use Case
- Protocol State
- Sequence
- Timing
- Communication

Unchanged
- Changed
- Significant Change
- New
Timing Diagram (New)

- Show change of classifier state with time in response to an event
- Objects on side (lifeline)
- States and timings inside
Information Flow Diagram (New)

• Shows flow of information between entities
• Wage information flows between Company and Employee
• High level diagram
  – Does not define information
  – Does not describe mechanisms
Collaborations

- Describe a pattern of collaboration between a number of roles
- Actual behaviour defined by interaction and class diagrams

- Can be used for
  - Design patterns
  - Architectural patterns
  - Business patterns
  - Key part of an application
What's New Summary

Diagram

Structural
- Class
- Object
- Package
- Deployment

Behavioural
- Activity
- State
- Use Case
- Protocol State

Component
- Composite Structure

Interaction
- Interaction Overview
- Sequence
- Timing
- Communication

Unchanged
- Changed
- Significant Change
- New
Component Diagram (New)

Use to define components
their interfaces and
connections to other
components
Components can be
subsystems, EJBs
Etc

Use these diagrams to
show how the system is
\textit{wired} together
Use a composite structure diagram to drill into a component

- Composite Structure defines instance level composition
What's New Summary

Diagram

Structural
- Class
- Object
- Package
- Activity
- Deployment
- Component
- Composite Structure
- Interaction Overview
- Sequence

Behavioural
- State
- Interaction
- Use Case
- Protocol State

Unchanged
Changed
Significant Change
New

Copyright 2003
Deployment Diagram (Minor Changes)

- As before. To show UML1 ‘component’ (jar file, config file etc) use class with <<artifact>> stereotype
- Maps artefacts to nodes
- Show deployment diagrams at instance or class level
• Connect nodes via a communications path
• Note the Node ‘supporting’ the deployed component
Extending UML 2
Tagged Values

• Can associate information with a model element via a property
  – Tag a value to the property and place close to the related model element
  – \{status = “under construction”, system analyst = “Fred”\}
  – Can use to pass detailed design information between roles
    • \{ standard algorithm = “Quick Sort”\}
Stereotypes

• Allow customisation of UML
• Extends a classifier to create a new one
• Optionally associate specific icon
• Many ‘pre-canned’ stereotypes
  – Uses, trace, refine …
• Analysis Stereotypes
  – Boundary, Control, Entity
  – Based on Model-view-control concept

Driver operations may call Ambulance operations

Analysis stereotypes
Constraints

- **Semantic condition / restriction on elements**
- **Many pre-canned constraints**
  - *Xor* constraint for associations
    - Indicates contract can *only* be owned by a person or a company
  - *Ordered* constraint for associations
Model Driven Architecture (MDA)
Whats MDA?

- **Model Driven Architecture**
  - Making UML models executable
  - Integration between different platforms by
    - Specifying a layered approach to architecture
  - May 2003 OMG published MDA Guide v1.0
  - Vision is
    - Automated compilation from the model
  - Implies two things
    - Increased semantic precision (UML 2)
    - Raise the modellers level of abstraction
      - Use models to manage and organise the complexity
• Architect must model and apply mapping rules to transform the layers – marking
• Architect selects profile most appropriate for the problem to Ensure efficient code generation
MDA -3 - Mapping it all Together
MDA – 4 – Wrapping it Up

• MDA Implies
  – Architects, Business Analysts, Developers have to raise their game and model….
  – Reference Architectures (profiles)
    • These are of limited availability today
    • Early adopters will be creating them
  – Get to grips with model reuse
  – Get to grips with the legacy estate that runs most large businesses
  – But to fulfil the promise of MDA even the PIM must be rigorously defined!

• MDA Aims to
  – Provide a vendor neutral approach to managing the typical heterogeneous environment
  – Provides a framework to help software engineers share information

• In Summary
  – It's early days …
  – MDA is for Early Adopters
UML 2 – Wrap Up

• UML 2 has a number of key objectives
  – Build in support for MDA (executable models)
  – More robust workflow/action modelling
  – Provide a model communication standard (Diagram Interchange Model)
  – Place UML within standard modelling framework (MOF – Meta Object Facility)
  – Provide real time support and support for common execution environments

• UML 2 has made significant changes to and added new diagrams

• But Remember the goal of it all is clear unambiguous communication
  – Do enough modelling to achieve that goal on your project in your organisation and then stop
UML 2 – The Diagrams

**Structural**
- Class
- Object
- Package
- Deployment
- Component
- Composite Structure

**Behavioural**
- Activity
- Use Case
- Behavioural State
- Protocol State
- Interaction Overview
- Sequence
- Timing
- Communication
- Information Flow