IBM European Rational Technical Conference
October 17–21 2005 - Düsseldorf

Describing Software Architectures with UML 2.0

Peter Eeles
peter.eeles@uk.ibm.com
Outline

What is Software Architecture?
- How is a Software Architecture described?
- Software Architecture and UML 2.0
- Summary
A Bit of Modern Software...

SC_MODULE(producer)
{
  sc_outmaster<int> out1;
  sc_in<bool> start; // kick-start
  void generate_data()
  {
    for(int i =0; i <10; i++)
    {
      out1 =i; //to invoke slave;
    }
  }
  SC_CTOR(producer)
  {
    SC_METHOD(generate_data);
    sensitive << start;
  }
}

SC_MODULE(consumer)
{
  sc_inslave<int> in1;
  int sum; // state variable
  void accumulate()
  {
    sum += in1;
    cout << “Sum = “ << sum << endl;
  }
  SC_CTOR(consumer)
  {
    SC_SLAVE(accumulate, in1);
    sum = 0; // initialize
  }
}

SC_MODULE(top) // container
{
  producer *A1;
  consumer *B1;
  sc_link_mp<int> link1;
  SC_CTOR(top)
  {
    A1 = new producer(“A1”);
    A1.out1(link1);
    B1 = new consumer(“B1”);
    B1.in1(link1);
  }
}

Can you spot the architecture?
…and its UML 2.0 Model
Breaking the Architecture....

SC_MODULE(producer)
{
sc_outmaster<int> out1;
sc_in<bool> start; // kick-start
void generate_data ()
{
for(int i =0; i <10; i++) {
out1 =i ; //to invoke slave;}
}
SC_CTOR(producer)
{
SC_METHOD(generate_data);
sensitive << start;}};
SC_MODULE(consumer)
{
sc_inslave<int> in1;
int sum; // state variable
void accumulate (){
sum += in1;
cout << “Sum = “ << sum << endl;}
SC_CTOR(consumer)
{
SC_SLAVE(accumulate, in1);
sum = 0; // initialize
};
SC_MODULE(top) // container
{
producer *A1;
consumer *B1;
sc_link_mp<int> link1;
SC_CTOR(top)
{
A1 = new producer(“A1”);
//A1.out1(link1);
B1 = new consumer(“B1”);
//B1.in1(link1);}};

Can you see where?
Breaking the Architecture….

Can you see it now?
What is Software Architecture?

  - Architecture is the **fundamental organization** of a system embodied in its components, their relationships to each other, and to the environment, and the **principles guiding its design and evolution.** [IEEE 1471-2000. “Architectural Description of Software-Intensive Systems”]

- **Software Architecture in Practice [Bass]**
  - The software architecture of a program or computing system is the **structure** or structures of the system, which comprise software elements, the externally visible **properties** of those elements, and the **relationships** among them.

- **UML 1.5**
  - [Architecture is] the organizational **structure** and associated **behavior** of a system. An architecture can be recursively **decomposed** into **parts** that interact through **interfaces**, **relationships** that connect parts, and **constraints** for assembling parts. Parts that interact through interfaces include classes, components and subsystems.
Outline

- What is Software Architecture?
- How is a Software Architecture described?
- Software Architecture and UML 2.0
- Summary
IEEE 1471-2000

Figure 1—Conceptual model of architectural description
Architecture description – Kruchten 4+1 views
Architecture description – RM-ODP
Models, Views and Diagram

- Logical View
  - Diagram
  - Design Model

- Implementation View
  - Diagram
  - Implementation Model

- Deployment View
  - Diagram
  - Deployment Model
Architecture and Modeling

- Software architectures are specified by models:
  - To architect is to model
- Software modeling languages used by software architects must have appropriate architectural modeling capabilities
- What are those capabilities?
Outline

- What is Software Architecture?
- How is a Software Architecture described?
- Software Architecture and UML 2.0
- Summary
UML 2.0 Specification

- UML 2.0 Superstructure
  - Defines the six structure diagrams, three behavior diagrams, four interaction diagrams, and the elements that comprise them

- UML 2.0 Infrastructure
  - Defines base classes that form the foundation not only for the UML 2.0 superstructure, but also for MOF 2.0

- UML 2.0 Object Constraint Language (OCL)
  - Allows setting of pre- and post-conditions, invariants, and other conditions

- UML 2.0 Diagram Interchange
  - Extends the UML metamodel with a supplementary package for graph-oriented information, allowing models to be exchanged or stored/retrieved and then displayed as they were originally
The UML 2.0 Superstructure Specification


This OMG document replaces the submission document (ad-03-04-01) and the Draft Adopted specification (ptc/03-07-05). It is an OMG Final Adopted Specification and is currently in the finalization phase. Comments on the content of this document are welcomed, and should be directed to issues@omg.org by September 8, 2003.

You may view the pending issues for this specification from the OMG revision issues web page http://www.omg.org/issues/, however, at the time of this writing there were no pending issues.

The FTF Recommendation and Report for this specification will be published on April 30, 2004. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.
UML 2.0 Diagrams

- **UML 1.5**
  - Structure
    - Class diagram
    - Object diagram
    - Component diagram
    - Deployment diagram
  - Behavior
    - Use case diagram
    - Activity diagram
    - Statechart diagram
    - Interaction
      - Sequence diagram
      - Collaboration diagram

- **UML 2.0**
  - Structure
    - Class diagram
    - Object diagram
    - Component diagram
    - Deployment diagram
    - Package diagram
    - Composite structure diagram
  - Behavior
    - Use case diagram
    - Activity diagram
    - State machine diagram
    - Interaction
      - Sequence diagram
      - Communication diagram
      - Interaction overview diagram
      - Timing diagram*

UML 2.0 Diagrams

Structure:
- Object Diagram
- Component Diagram
- Deployment Diagram
- Package Diagram
- Class Diagram
- Composite Structure Diagram

Behavior:
- Use-Case Diagram
- Timing Diagram
- Activity Diagram
- State Machine Diagram
- Sequence Diagram
- Communication Diagram
- Interaction Overview Diagram
Frames

- Each diagram has a frame, a content area and a heading
- The frame is a rectangle and is used to denote a border
  - Frame is optional
- The heading is a string contained in a name tag which is a rectangle with cut off corners in the upper left hand corner of the frame
  - Format `<kind> <name> <parameters>`
  - `<kind>` can be activity, class, component, interaction, package, state machine, use case
Frame

package CP

package P

C1
C2

C1
C2

class Car

rear : Wheel [2]
ade

e : Engine
Class Diagram

- Object Diagram
- Component Diagram
- Deployment Diagram
- Package Diagram
- Composite Structure Diagram
- Timing Diagram
- Interaction Overview Diagram
- Use-Case Diagram
- Activity Diagram
- State Machine Diagram
- Sequence Diagram
- Communication Diagram

Structure

Behavior
Class Diagram

- **Customer**
  - name: String
  - phone: String
  - add(name, phone)
  - owner (association, role names)
  - purchased (association, role names)

- **Reservation**
  - date: Date
  - generalization

- **Subscription Series**
  - series: Integer
  - constraint (xor)

- **Individual Reservation**
  - 0..1

- **Ticket**
  - available: Boolean
  - sell(c:Customer)
  - exchange()

- **Show**
  - name: String
  - show
  - 1..*
  - performances

- **Performance**
  - date: Date
  - time: TimeOfDay
UML 2.0 Interface Notation

- PrintServer
  - Provided Interface: SubmitJob, CheckStatus, SetPrintProperties
  - Required Interface: TransmitData

- FinancialPlanner
- ManualPriceEntry
- MutualFundAnalyzer
- QuoteQuery

- UpdatePrices
  - Required Interface: UpdatePrices
  - Provided Interfaces: FinancialPlanner, ManualPriceEntry, MutualFundAnalyzer, QuoteQuery
Object Diagram
Hierarchy of UML 2.0 Classifiers (simplified)

- **Classifier**
- **Class**
- **Structured Class**
  - Internal structure
  - Ports
  - Connectors

- **Component**
Component Diagram
Deployment Diagram

- Object Diagram
- Component Diagram
- Package Diagram
- Class Diagram
- Use-Case Diagram
- Activity Diagram
- State Machine Diagram
- Sequence Diagram
- Communication Diagram
- Composite Structure Diagram
- Timing Diagram
- Interaction Overview Diagram

Structure

Behavior
Deployment Diagram

IBM European Rational Technical Conference 2005

Web Server
- «executable» presentationLogic.exe

Application Server
- «executable» businessLogic.exe

Database Server

Workstation

Deployment Diagram

Web Server
- «deploy»
  - «executable» presentationLogic.exe

Application Server
- «deploy»
  - «executable» businessLogic.exe

Database Server

Workstation

Deployment Diagram

Web Server
- «ADS Mapping»
  - «RequestContainer»

Application Server
- «ADS Mapping»
  - «TransactionContainer»
Package Diagram

Object Diagram

Component Diagram

Deployment Diagram

Package Diagram

Class Diagram

Use-Case Diagram

Activity Diagram

State Machine Diagram

Sequence Diagram

Communication Diagram

Timing Diagram

Composite Structure Diagram

Interaction Overview Diagram

Behavior

Structure
Package Diagram

- Definition
  - A structure diagram whose content is primarily packages and their relationships. There is no rigid line between the different kinds of structure diagrams, so the name is merely a convenience without semantic significance.

- Structure Diagram
  - A diagram that describes the static structure of a system, as opposed to its dynamic behavior. There is really no rigid line between different kinds of structure diagrams, although diagrams may be named according to the major kind of element they contain, such as class diagram, interface diagram, or package diagram.
Package Diagram

IBM European Rational Technical Conference 2005

Planning

Publicity

Scheduling

Box Office

Customer Records

Ticket Sales

Ticket Records

Operations

Purchasing

Accounting

Payroll
Composite Structure Diagram

Object Diagram
Component Diagram
Deployment Diagram
Package Diagram

Class Diagram
Use-Case Diagram
Activity Diagram
State Machine Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram

Structure

Behavior

Composite Structure Diagram

Timing Diagram

Diagram

Diagram
The Collaborating Peers Structural Pattern

- Two (or more) components that collaborate to achieve some greater objective
  - Each can exist independently of the other

- Connectors indicate only valid couplings between components
  - Unconnected components cannot affect each other directly
  - Explicit specification of architectural constraints
Aren’t Class Diagram Sufficient?

- No!
  - For example, because they abstract out certain specifics, class diagrams are not suitable for performance analysis
- Need to model structure at the instance level

Same class diagram describes both systems!
Are Object Diagrams What We Need Then?

- No!
  - Object diagrams represent “snapshots” of some specific system at some point in time
  - They can only serve as examples and not as general architectural specifications

- Need a way of talking about “prototypical” instances across time
Representing Collaborating Peers

- Internal structure represents a collaboration of roles communicating via connectors
- A role can represent an instance or something more abstract
Composite Structure Diagram

- A diagram that shows the internal structure (including parts and connectors) of a structured classifier or a collaboration. There is no rigid line between a composite structure diagram and a general class diagram.
The **Port** Structural Pattern

- Distinct interaction points of an object for multiple, possibly simultaneous collaborations
- Ports allow an object to distinguish between different external collaborators without direct coupling to them
In general, a port can interact in both directions.
Shorthand Notation

- **DBserver** to **clientPort**
- **DBclient** to **clientPort**
- **adminPort**

**DataBase**
Connectors and Protocols

- Ports can also be joined by connectors.
- Connectors can be constrained to by a collaboration protocol:
  - Static type checking for dynamic flow violations are possible.
  - Eliminates a major source of “integration” errors.
Internal Parts and Ports of a Component
UML 2.0 Profile for Software Services

Use-Case Diagram
Activity Diagram

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

Structure

Behavior
Activity Diagram

BoxOffice::ProcessOrder

- set up order
  - guard condition [single order]
  - decision (branch)
    - subscription
    - synchronization bar (fork)
      - assign seats
      - award bonus
      - charge credit card
        - debit account
          - synchronization bar (join)
            - merge (unibanch)
              - mail packet
                - end of activity
Activity Diagram UML 2.0 Changes

- Data flow noted with a solid line
- Small sub-nodes (pins) added to show inputs/outputs
- Activities represented as round-cornered rectangles (as opposed to round-sided rectangles)
- Chevron control icons are replaced with pentagons
- Partitioning capabilities
Partitioning capabilities

- Seattle
  - Order Department
    - Receive order
    - Fill order
    - Ship order
  - Accounting
    - Send invoice
    - Invoice
  - Customer
    - Make payment

- Reno
  - Order Department
    - Receive order
    - Fill order
    - Ship order
    - Close order
  - Accounting
    - Accept payment
  - Customer
  - Order
  - Payment

Customer

Invoice

Make payment

Accept payment

Send invoice

Send payment

Ship order

Close order

Make payment

Customer

Invoice

Make payment

Accept payment

Send invoice

Send payment

Ship order

Close order

Make payment
State Machine Diagram

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

Structure

Behavior

Overview
State Machine Diagram

- **Purchasing**
  - exit / eject card
  - submachine reference
  - initial state

- **Identify**
  - fail / explicit exit

- **Selecting**
  - pick (seat) / add to selection (seat)
  - normal exit / reset selection

- **Confirming**
  - push “confirm”

- **Selling**
  - entry / sell ()

- **Idle**
  - insert card
  - completion / transition

- **Push “cancel”**

- **Outer transition aborts internal activity**

- **Push “resume”**

- **Push “buy” event**

- **Completion transition**

- **Final state**
Sequence Diagram

Structure

Object Diagram
Component Diagram
Deployment Diagram
Package Diagram
Class Diagram
Use-Case Diagram
Activity Diagram
State Machine Diagram
Composite Structure Diagram
Timing Diagram
Interaction Overview Diagram

Behavior

Sequence Diagram

UML Modeling Language
Sequence Diagram

**sd ATM-transaction**

- **client:**
  - insertCard

- **atm:**
  - **ref** CheckPin
    - [chk= OK]
    - **alt**
      - **ref** DoTransaction
        - error(badPIN)
        - [else]

- **dbase:**

**sd CheckPin**

- **client:**
- **atm:**
  - askForPIN
  - data(PIN)
  - check(PIN)
- **dbase:**
  - result(chk)
  - result(chk)

**Combined (in-line) Fragment**
Combined Fragment Types

- **Alternatives (alt)**
  - choice of behaviors – at most one will execute
  - depends on the value of the guard (“else” guard supported)

- **Option (opt)**
  - Special case of alternative

- **Break (break)**
  - Represents an alternative that is executed instead of the remainder of the fragment

- **Parallel (par)**
  - Concurrent (interleaved) sub-scenarios

- **Negative (neg)**
  - Identifies sequences that must not occur

- **Critical Region (region)**
  - Traces cannot be interleaved with events on any of the participating lifelines

- **Assertion (assert)**
  - Only valid continuation

- **Loop (loop)**
  - Optional guard: [<min>, <max>, <Boolean-expression>]
  - No guard means no specified limit
Communication Diagram

- Object Diagram
- Component Diagram
- Deployment Diagram
- Package Diagram
- Class Diagram
- Composite Structure Diagram
- Use-Case Diagram
- Timing Diagram
- Activity Diagram
- State Machine Diagram
- Sequence Diagram
- Interaction Overview Diagram

Structure

Behavior

Communication Diagram
Communication Diagram

sd Reserve Flights

:Client

:Flight Manager

1 approve flights
1.3 confirm flights
2 pay
2.1 receipt
1.2 confirm flights
1.1 approve flights

:Flight DB
Interaction Overview Diagram
Interaction Overview Diagram
Timing Diagram

Structure

Behavior

Object Diagram
Component Diagram
Deployment Diagram
Package Diagram

Class Diagram
Use-Case Diagram
Activity Diagram
State Machine Diagram
Sequence Diagram
Communication Diagram

Composite Structure Diagram
Timing Diagram
Interaction Overview Diagram
Summary

- Software architectures are specified by models
- UML 2.0 provides a convenient set of modeling concepts suitable for directly capturing most interesting architectural patterns
- UML 2.0 can therefore support a variety of architecture description languages (ADLs)
Questions
Thank You