Object-orientation as Composite Structure:
(On-to)Logical Object-orientation

Conrad Bock,
U.S. National Institute of Standards and Technology

Charles Galey
Lockheed Martin
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary
Behavior as Composite Structure

Presentation Stack

Onto State Machines
(next meeting)

Onto Interactions
(ad/18-06-11)

Onto OO
(this one)

Onto Behavior Basics
(ad/2018-03-02)
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary
General Problem

- UML has three behavior diagrams.
  - Activity, state, interaction.
- Very little integration or reuse between them.
  - Three underlying metamodels.
  - Three representations of temporal order.
- Triples the effort of learning UML and building analysis tools for it.
General Solution

- Treat behaviors as assemblies of other behaviors.
  - Like objects are assemblies of other objects.

- Assembly = UML internal structure
  - Pieces represented by properties.
  - Put together by connectors.

- Put all behavior diagrams on the same underlying behavior assembly model.
Behaviors as Composite Structure

**State Diagram**

- Traction-detector (d1)
- Braking-force modulator (m1)

**Sequence Diagram**

- detTrkLos() for detecting track loss
- sendSignal() for sending signal
- modBrkFrc(traction_signal:boolean) for modulating braking force
- sendAck() for sending acknowledgment

**Activity Diagram**

- Prevent lockup (act PreventLockup)

**State Machine Diagram**

- Gripping
- Slipping
- LossOfTraction
- RegainTraction
“Things” that occur in time
- Eg, taking a picture, focusing, etc.
- Not “behaviors”, “actions”, etc.

Real, Simulated, or Desired Things Being Modeled (M0)

Not instance specs.

Focus 3/15/09 10-11amET:

TakePicture 3/15/09 10-12pmET:

Shoot 3/15/09 11-12pmET:
Behavior: What’s in Common?

- They happen before or during each other.
  - Construct M1 library for this.
  - Use it to classify things being modeled.
Specialize library classes and subset/redefine library properties.
Capture M1 patterns in M2 elements.
- Tools apply patterns automatically.
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary
Interactions Problem

Interactions Problem

Object Flow

Activity

SysML Internal Block Diagram

Item Flow

Message
Interactions Requirements

1. Between things that **outlive interactions**.
   - Objects have many interactions over time.
   - Not just between steps in an activity.

2. Interactions are **reusable and composable**.
   - The same kind of interaction might be used in many user models and
   - contain many other interactions ordered in time.

3. Interacting objects have “**mailboxes**”.
   - Things being exchanged leave and arrive at specified places in the interacting objects.
   - Aka, output/inputs.
Transfers (M1)

Standard Model Library

Model (M1)

User Model

Things Being Modeled (M0)

Behavior Occurrence

Transfer

Product Transfer

Any Thing

Stove234:

Product Transfer 3/15/09 10-12pmET:

Product

Store654:

John’sHouse:

Product

John’s House
Interactions (M2)

M1 property at tail of arrow is value of M2 property at head of the arrow. *Not instance links*
Transfers Along Connectors?

- Connectors are typed by associations.

- But transfers are behaviors.

- Connectors are typed by associations.

- But transfers are behaviors.
Interaction = Behavior & Association

- Associations and behaviors both have objects that participate in them.
  - Associations link their participants.
  - Behaviors involve their objects.
    - Interactions have lifelines.
    - Activities have object nodes, partitions, etc.
    - Behaviors have parameters.

- Interactions are behaviors that are also associations between their participants.
Links (M1) & Associations (M2)

Metamodel (M2)

Standard Model Library

Model (M1)

User Model

Things Being Modeled (M0)

Class

Association

Property

 owned Property

participant Property

[subsets]

*Not instance links*

Model (M1)

Anything

Camera

Controller

Camera 34 : linkedCam

Camera

Controller

Linked Thing

{non-unique}

[2..*]

{subsets}

linkedCon : Controller {redefines linkedSource}

linked Cam : Camera {redefines linkedTarget}

Linked Thing

Linked Con

M1 property at tail of arrow is value of M2 property at head of the arrow.

User Model

Camera

Controller

Link

*\[\text{linkedCam}\]

\[\text{linkedCon}\]

\[\text{conCam}\]

\[\text{camCon}\]

\[\text{Cam} \scriptstyle{\text{era}}\scriptstyle{\text{34}} : \text{linkedCam}\]

\[\text{Cntrl} \scriptstyle{\text{12}} : \text{linkedCon}\]

\[\text{L} \scriptstyle{\text{ink}} \scriptstyle{\text{251}} : \text{conCam}\]
Transfers as Links (M1)

Standard Model Library

Model (M1)

User Model

Things Being Modeled (M0)
Interaction Participants (M2)

**Metamodel (M2)**

- **Behavior**
  - **Association**
  - **Property** (redefines)
  - **Interaction**
  - **Participant**
  - **Property**
  - **Class**
  - **Property**
  - **any**

**Model (M1)**

- **Link**
  - **Behavior Occurrence**
  - **Transfer**
  - **Source Thing**
  - **Target Thing**
  - **Any Thing**
  - **Property**
  - **Transfered Thing**
  - **Product Transfer**
  - **Product**

**Standard Model Library**

- **Model**
  - **User Model**

M1 property at tail of arrow is value of M2 property at head of the arrow. *Not instance links*
Connectors Reusing Interactions

Metamodel (M2)

User Model (M1)

Things Being Modeled (M0)

Connectors Reusing Interactions

Owned Property

Connector

Interaction

Association

Class

Property

Flow

DeliverProduct :

pickupFrom :

deliverTo :

Product Delivery 3/15/09 9-1pm ET :

pickupFrom

deliverTo

Store654:

John’s House:

Stove234:

Product Transfer 3/15/09 10-12pm ET :

pt : ProductTransfer

pt

transferredThing

{(subsets)}

{(redefines)}

*
Flow Steps

Metamodel (M2)

- Class
- Property
- Connector
- Association
- Behavior
- Step
- Succession
- Interaction
- Flow

Model (M1)

- CapturePicture
- Command
- Spacecraft
- Picture
- Standard Model Library

User Model

- fcntl: Flight Control
- fdb: Flight Database
- sc: Spacecraft
- : HappensBefore
- : Confirmation
- : HappensBefore
- Transfer

Behavior Occurrence

happens Before

(redefines)

earlierStep

laterStep

happensBefore

Flow Steps

Model (M1)

- User Model
- Standard Model Library

Metamodel (M2)

- Class
- Property
- Connector
- Association
- Behavior
- Step
- Succession
- Interaction
- Flow

(redefines)

ownedStep

earlierStep

laterStep

happensBefore

Before

: HappensBefore

Standard Model Library

happens Before

(redefines)
Flows & Out/Inputs (OF)

Metamodel (M2)

- **Class**
  - Item Flow
- **Property**
  - sourceOutputProperty
  - targetInputProperty
  - `typeOfThing Flowing`
  - `[1..*] {ordered, non-unique}`

Model (M1)

- **TakePicture** Activity
  - step1: Focus
    - `out xrsl: Exposure`
  - step2: Shoot
    - `in xfs: Exposure`

Instances (M0)

- **Focus Occ 1**: `out xrsl = Exp123`
- **Shoot Occ 1**: `in xfs = Exp123`

M1 property at tail of arrow is value of M2 property at head of the arrow.
*Not instance links*

TakePicture 3/15/09 10-12pmET:
Flows & Out/Inputs (FP)

Metamodel (M2)

Model (M1)

CapturePicture : Interaction

M1 property at tail of arrow is value of M2 property at head of the arrow. *Not instance links*
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - **OO behavior, requirements**
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary
OO Problems in UML/SysML

- Encapsulated and “surfaced” behaviors modeled differently.
  - Namespace ownership for encapsulated behaviors (methods).
  - Operations for surfaced behaviors.
- Method specialization (“override”) doesn’t use generalization / inheritance.
Interface (service “bundles”)
- Missing supported interactions.
  - Expected order of operation calls, signal receipts, flowing property values.
- Redundantly specified on both ends of interactions (eg, conjugation).
- Need ports to distinguish interfaces uses.
- Redundant model of behavior abstraction
  - Specify input/outputs of surfaced behaviors (ie, they abstract those behaviors).
  - But UML interface realization not generalization.
1. Behavior encapsulation
   – “Surfaced” behaviors (no steps)

2. Behavior inheritance

3. Protocols
   – Expected order of using surfaced behaviors.
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary
Properties of Objects for Behavior Occurrences

- Values of these properties are executions (occurrences, M0 instances) of behaviors.
  - For example, classifier behavior executions.
Behaviors not encapsulated.
- Controller specifies “how” picture is taken.
- Compare to activity partitions.

Controller should only specify inputs and outputs for camera and disk behaviors.
Encapsulating Behaviors

Model (M1)

Camera&DiskController

- External behavior properties (operations)
  - Types only “expose” inputs and outputs.
  - Have same executions (equal values) as internal behavior properties (methods).
- (Not ports)
External Behaviors

Metamodel (M2)

- Generalization
- Class
- Behavior
- Implements
- Surface Behavior
- TakePicture

Model (M1)

- TakePicture
  - step1: Focus
  - step2: Shoot

Things Being Modeled (M0)

- TakingPic1 3/15/09 2pmET:
- TakingPic2 4/19/09 1amET:
- TakingPic3 2/5/10 8pmET:
Operations

Metamodel (M2)

Model (M1)

Things Being Modeled (M0)
Behavior Invocation

“Calls” are behaviors that constrain surrounding successions and item flows.
  - Specify whether to wait for return (synchronous/asynchronous calls).
  - Have no steps (“no-ops”).
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary
Specializing Methods

- Not OO “overriding”:
  - Specialized methods cannot remove inherited elements, only specialize them.
  - Use general methods for commonality among implementations
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary
OO View of Interactions

- Objects support interactions by providing “services” (including data).
  - UML added services required of other objects.

- Object models (classes) typically do not specify the interactions they support.
  - Only services “surfaced” to the outside.
  - Except for UML’s protocol state machines.
OO & Interaction Approaches

Object

Ports & Interfaces

Object

Ports & Interfaces

Object

Participant Role

Interaction

Object

Participant Role

Object
Protocols for Using Operations

- **Protocol:**
  - Power must be turned on before taking picture.
  - Multiple pictures can be taken.
  - Power must be turned off after the last picture is taken.
Protocol as Interaction

Model (M1)

CameraTakePicture : Interaction

device : Camera

: PowerOn

: Take Picture

Location

Picture

callPowerOn :

: CallPowerOn

: CallTakePicture

controller :

: CallPowerOff

: CallPowerOff

: PowerOff
Protocol as Interaction (M2)

Metamodel (M2)

Class protocol Interaction

Metamodel

Camera Model (M1)

Camera protocol CameraTakePicture

CallPowerOn : Location

CallPowerOff : Picture

callPowerOn : Location

callPowerOff : Picture
Using Interaction Protocols

Model (M1)

SLRPhotography : Interaction

- cam : SLRCamera
- photographer : Person

CameraTakePicture

- device : Camera
- controller : 
- callPowerOn : Location
- Picture
- callPowerOff :
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary
Could use as interaction participant types.
OO Protocols

- Defined without external objects.
Conjugation

- UML required operations = service requests sent to external objects.
**OO Inputs and Outputs**

**Metamodel (M2)**
- Generalization
- Class
- Behavior
- Property

**Model (M1)**
- TakePicture
- step1: Focus
- step2: Shoot
- ^subject: Location
- ^result: Picture

**Things Being Modeled (M0)**
- TakingPic1: 3/15/09 2pmET
- TakingPic2: 4/19/09 1amET
- TakingPic3: 2/5/10 8pmET

Notation:
- I/O only.
- *Not instance links*
Multiple OO Interfaces

- Connector uses both interfaces or one?
  - If one, which?
Port for Each OO Interface

- Typed by interfaces, not operations.
- Raises questions:
  - Are ports separate from objects they’re on?
  - If separate, are they internal or external parts?
  - Tied up an entire SysML RTF.
Multiple Interaction Protocols

- Connectors typed by different interactions.
  - Ports not needed.
Multiple Ports for Same Interface

- Object can interact differently based on port used.
  - Better to define with separate interactions.
- If same interaction, use correlation (BPMN).
- Not possible with interaction protocols.
Overview

- RoadMap
- Motivation
  - Behavior, review
  - Interactions, review
  - OO behavior, requirements
- OO Behavior Solution
  1. Behavior encapsulation
  2. Behavior inheritance
  3. Protocols (interaction and OO)
- Summary
Summary

- **Unify OO behavior using**
  - Properties for operations and methods
  - Inheritance for “overriding” methods.
- **Simplify protocol modeling with**
  - Interactions instead of OO interfaces & ports.
- **Speeds learning and analysis integration.**
More Information

- Intro to Behavior as Composite Structure
- Interaction as Composite Structure
- Additional slides
  - Starts with onto, includes interactions.
- Paper: [http://dx.doi.org/10.5381/jot.2011.10.1.a3](http://dx.doi.org/10.5381/jot.2011.10.1.a3)
- Application to BPMN: [http://conradbock.org/#BPDM](http://conradbock.org/#BPDM)
- KerML: Contact Chas Galey [charles.e.galey@lmco.com](mailto:charles.e.galey@lmco.com)