zühlke
DIE DENKFABRIK.

UML with Action Semantics

Concepts, Application and Implications

Milan Ignjatovic
Software Engineering Consultant
Software Engineering Trainer
Zuehlke Engineering AG

Agenda

Part 1: What is UML with Action Semantics?

Part 2: Overview of Behavioural Modelling

Part 3: The Action Metamodel

Part 4: The Action Package

Part 5: Object Action Language

Part 6: Pathfinder Solutions Action Language – PAL

Part 7: Live Demonstration

Part 8: Implications and Summary

Part 9: Q/A

Before we start

Newton as seen by Blake

Part 1

What is UML with Action Semantics?

Visual intelligence

Kiczales:

- „The way we visualize code doesn't do much to use all we've learned about how to ***use form to reflect function***. This is critical, because most of the brain's cortex is visual"

Half of the cerebral cortex is devoted directly or indirectly to vision.

Hoffman: Visual Intelligence

- What does it mean to loose a critical aspect of visual intelligence? The story of Mr. P.

Q: what about Euler?

- What kind of model did he build? Language? Notation?
- How did he use the model? Was he model-driven?
- What if he was born blind or never gone blind at all?

What is UML with action semantics

Action - fundamental unit of computational behaviour

Action semantics are based on proven concepts from computer science

Action semantics remove assumptions about specific computing environments in user models:

- execution engines, PLs, implementation details
- do not require specification of software components, tasking structures or forms of transfer of control
- yet allows modellers to produce executable specifications

Action semantics have no normative notation

- OAL, PAL are concrete products and define own syntax

Open the eye of reality: layman‘s dream (Jacobson)

Complete specification available in UML 1.5 / Sept 2002

Terminology: xUML ≡ executable UML ⇒ UML with Action Semantics

How it works: xUML?

You capture and formalize knowledge

- Define the behaviour of the model in sufficient detail so that it can be executed

Use the model is like code

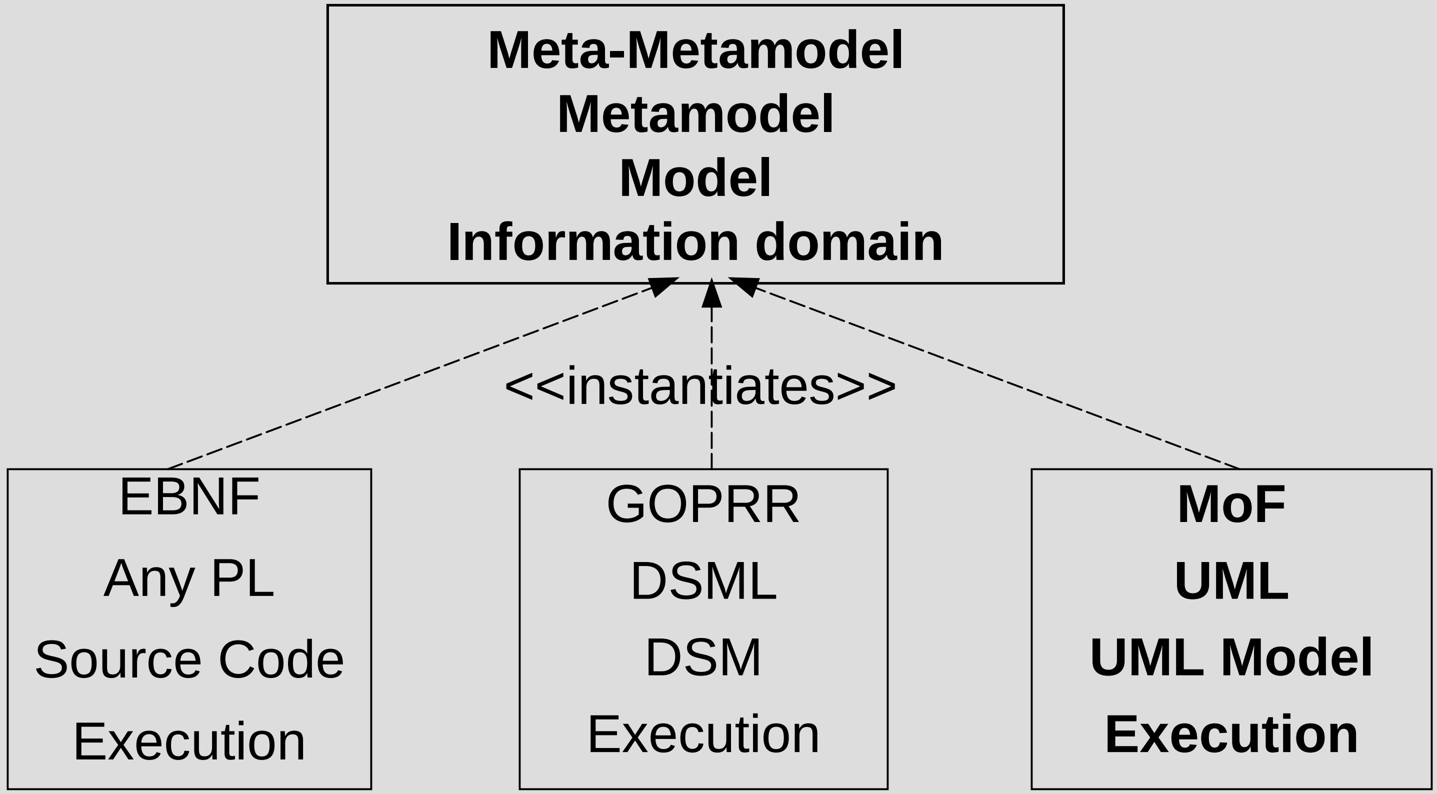
To get the running system: use the model compiler to compile several executable UML models each of which captures a single cross-cutting concern:

- analysis models
- design model i.e. design policies e.g. design patterns
- base mechanisms e.g. communication models
- ...

xUML models define the minimal model required to show how a domain operates in the context of problem*

xUML delivers executable analysis models

Meta-stacks



Meta-Metamodel
Metamodel
Model
Information domain

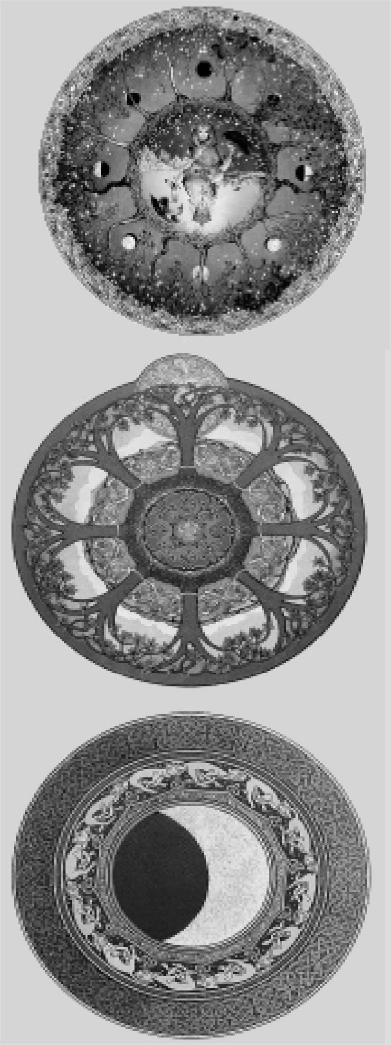
<<instantiates>>

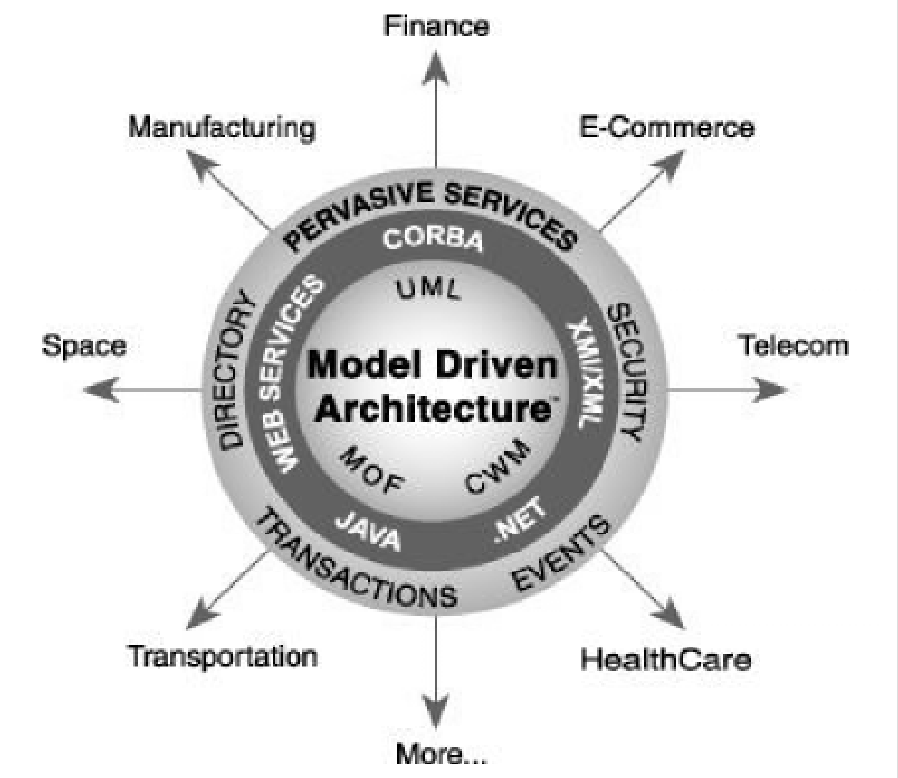
EBNF
Any PL
Source Code
Execution

GOPRR
DSML
DSM
Execution

MoF
UML
UML Model
Execution

xUML is the foundation for MDA





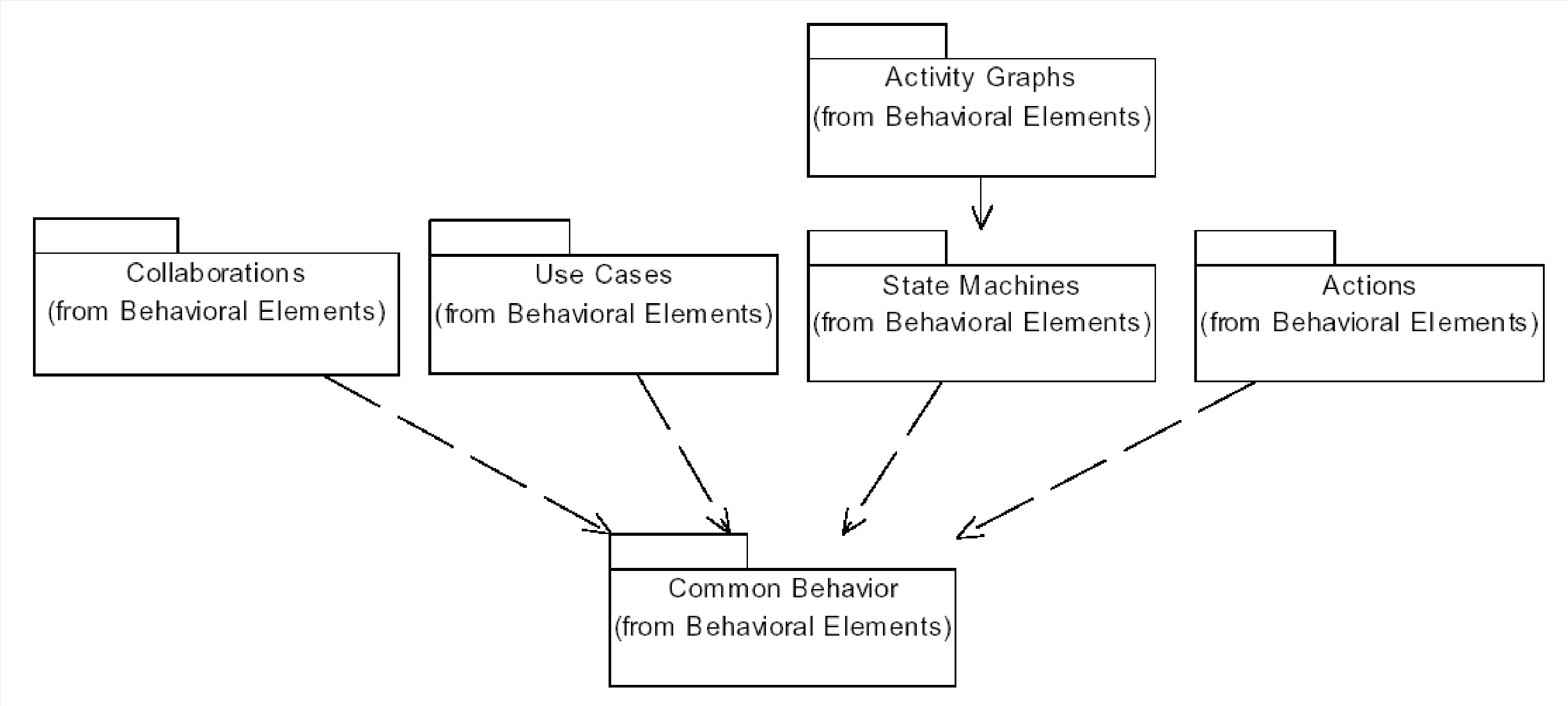
www.zuehlke.com

zühlke

Part 2

Overview of Behavioural Modelling

Behavioural elements


Activity Graphs
(from Behavioral Elements)
Collaborations
(from Behavioral Elements)
Use Cases
(from Behavioral Elements)
State Machines
(from Behavioral Elements)
Actions
(from Behavioral Elements)
Common Behavior
(from Behavioral Elements)

Use Cases and xUML

Focus of activities is moving upwards, to the front of the development process i.e. to analysis

Provide a foundation for modelling

- Identify domain ontology and emerging phenomena

Our objective here is to understand enough about the domain in order to build *executable models*

- Sky: doTrainSimulation

- Kite: LoadTrain, PositionTrain, StartSimulation

- Sea: Interact (triggered) by a single actor

- Mud: Complex UC hierarchies ending in technology details

UCs provide a source for test cases

***Beware: UCs may lead to poor abstractions if applied literally**

Part 3

The Action Metamodel

Actions: pins

An action takes a set of inputs and converts them into a set of outputs

Input pins

- hold values to be consumed by the action

Output pins

- hold values generated by the action

Pins are type conform

- The type of the output pin is the same as or is a descendant of the type of the input pin

Fan out of output pins is allowed

No fan in of input pins is possible

Actions: data flow, control flow

A data flow sequences execution of two actions by carrying data between them i.e. provides implicit sequencing

- A data flow has source and destination pins
- Output pins of one action are input pins of some other action

A control flow defines a sequencing dependency between two actions i.e. provides explicit sequencing

- The successor action of the flow may not execute until the predecessor action has completed execution

The specification maximises action concurrency

- it treats all actions as executing concurrently unless explicitly sequenced by a flow of data or control

Primitive actions, procedures

Primitive actions do not contain any subactions i.e. nested actions

Procedure is an action container: a set of actions within a model e.g. body of a method

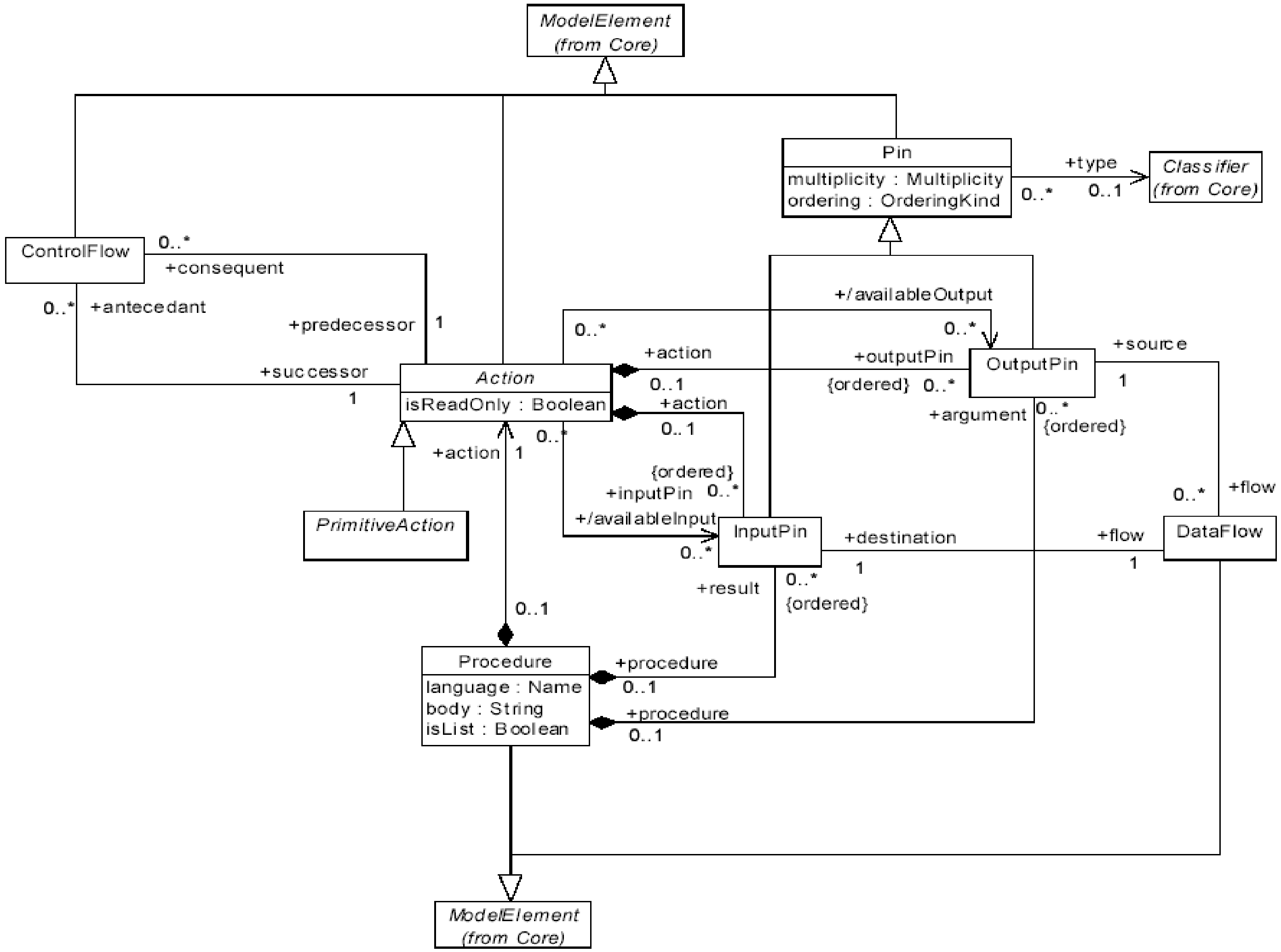
Procedure provides a context for action execution

Procedure takes a single object as argument and produces a single reply object as result

Multiple arguments or results possible i.e. represented as object attributes

May be attached to methods, state machines

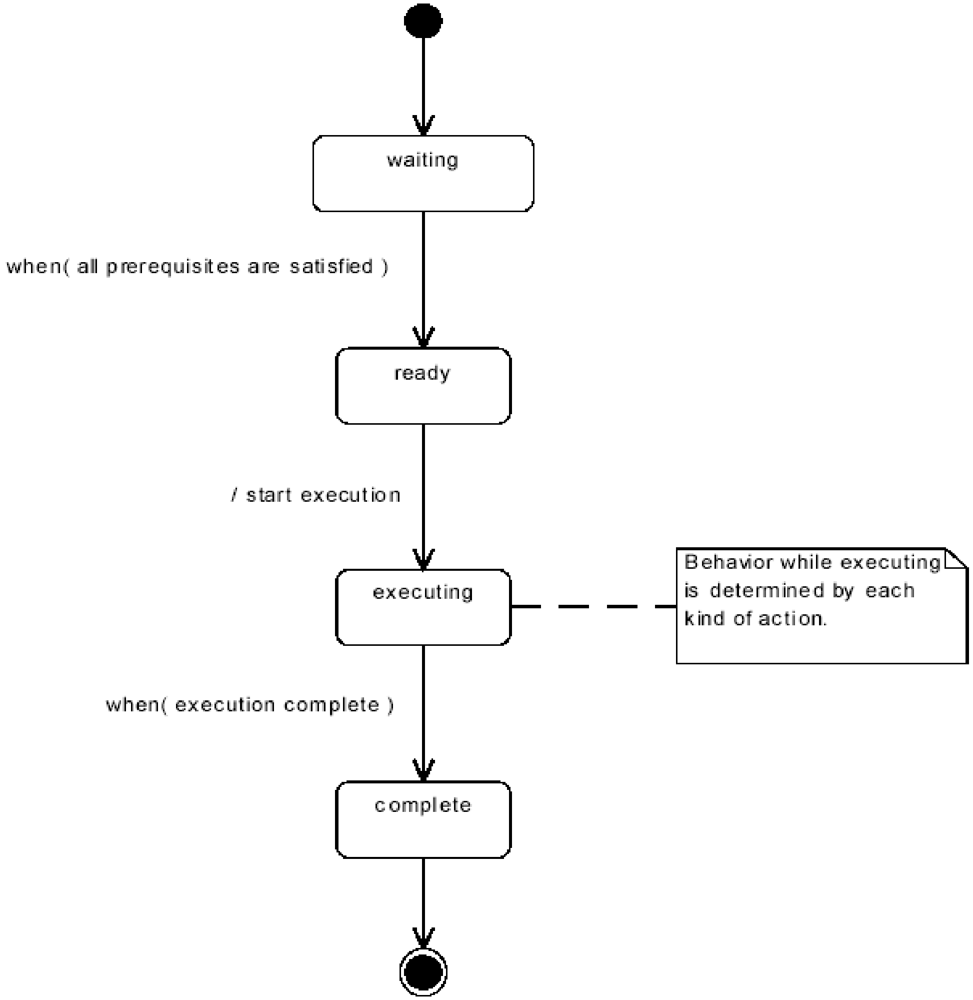
Action foundation model



www.zuehlke.com

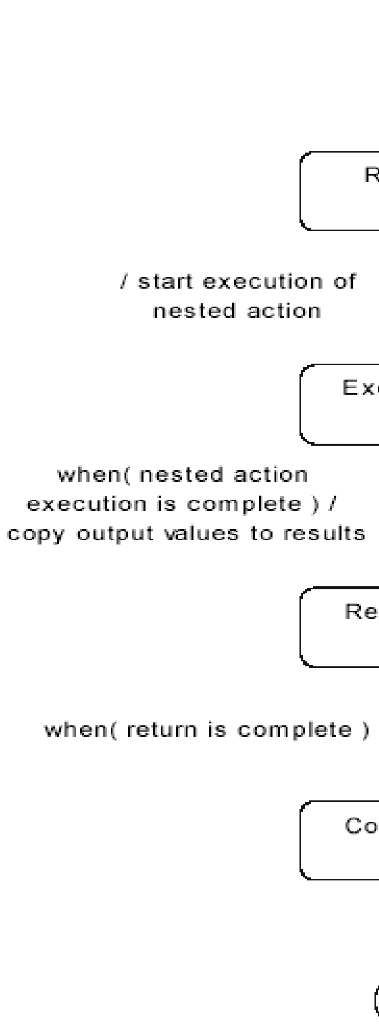
zühlke

Life cycle for action execution


waiting
when(all prerequisites are satisfied)
ready
/ start execution
executing
Behavior while executing is determined by each kind of action.
when(execution complete)
complete

zühlke

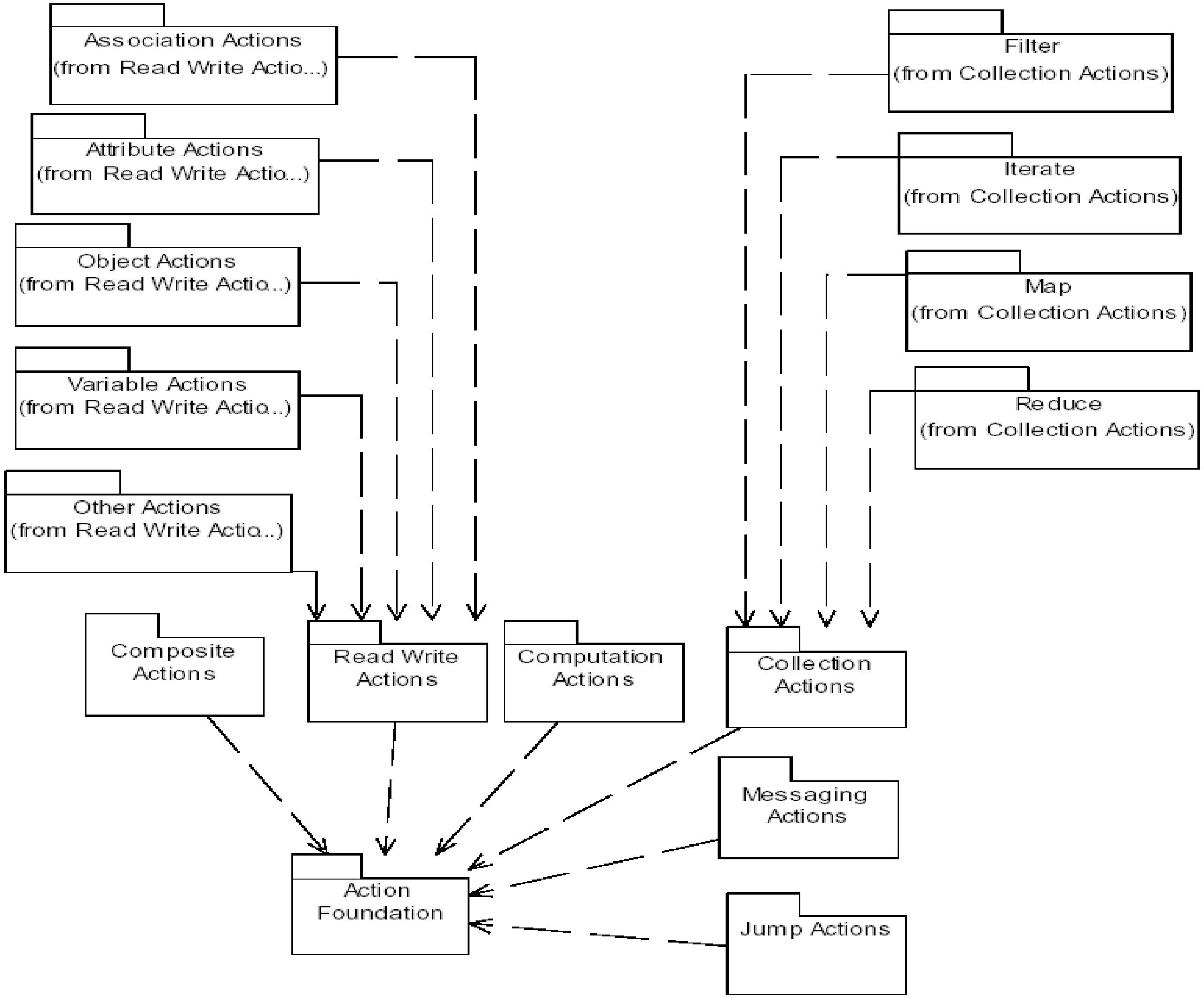
Life cycle for procedure execution


Ready
/ start execution of nested action
Executing
when(nested action execution is complete) / copy output values to results
Returning
when(return is complete)
Complete

Part 4

The Action Package

UML: Action package


Association Actions
(from Read Write Actio...)
Attribute Actions
(from Read Write Actio...)
Object Actions
(from Read Write Actio...)
Variable Actions
(from Read Write Actio...)
Other Actions
(from Read Write Actio...)
Filter
(from Collection Actions)
Iterate
(from Collection Actions)
Map
(from Collection Actions)
Reduce
(from Collection Actions)
Composite Actions
Read Write Actions
Computation Actions
Collection Actions
Messaging Actions
Jump Actions
Action Foundation

www.zuehlke.com

UML: Kinds of actions

New Data Types may be defined using metamodel Data Types e.g. UnlimitedInteger

- defines a data type whose range is the nonnegative integers augmented by the special value “unlimited”.
- used for the upper bound of multiplicities
- discussion of metamodel Data Types is beyond scope

Read and write actions

- variables, attributes, links

Composite actions

- group, conditional and loop actions

Computation actions

- Math is N/A, left to the implementation to define as needed
- ApplyFunctionAction, CodeAction, MarshalAction...

UML: More actions

Collection actions: contain a subaction, an embedded action that is executed once for each element in the input collection:

- Iterate: applies a subaction to each of the elements in a collection repeatedly within a loop
- Filter: selects a subset of the elements in a collection into a new collection
- Map: action applies a subaction in parallel to each of the elements in a collection

Messaging

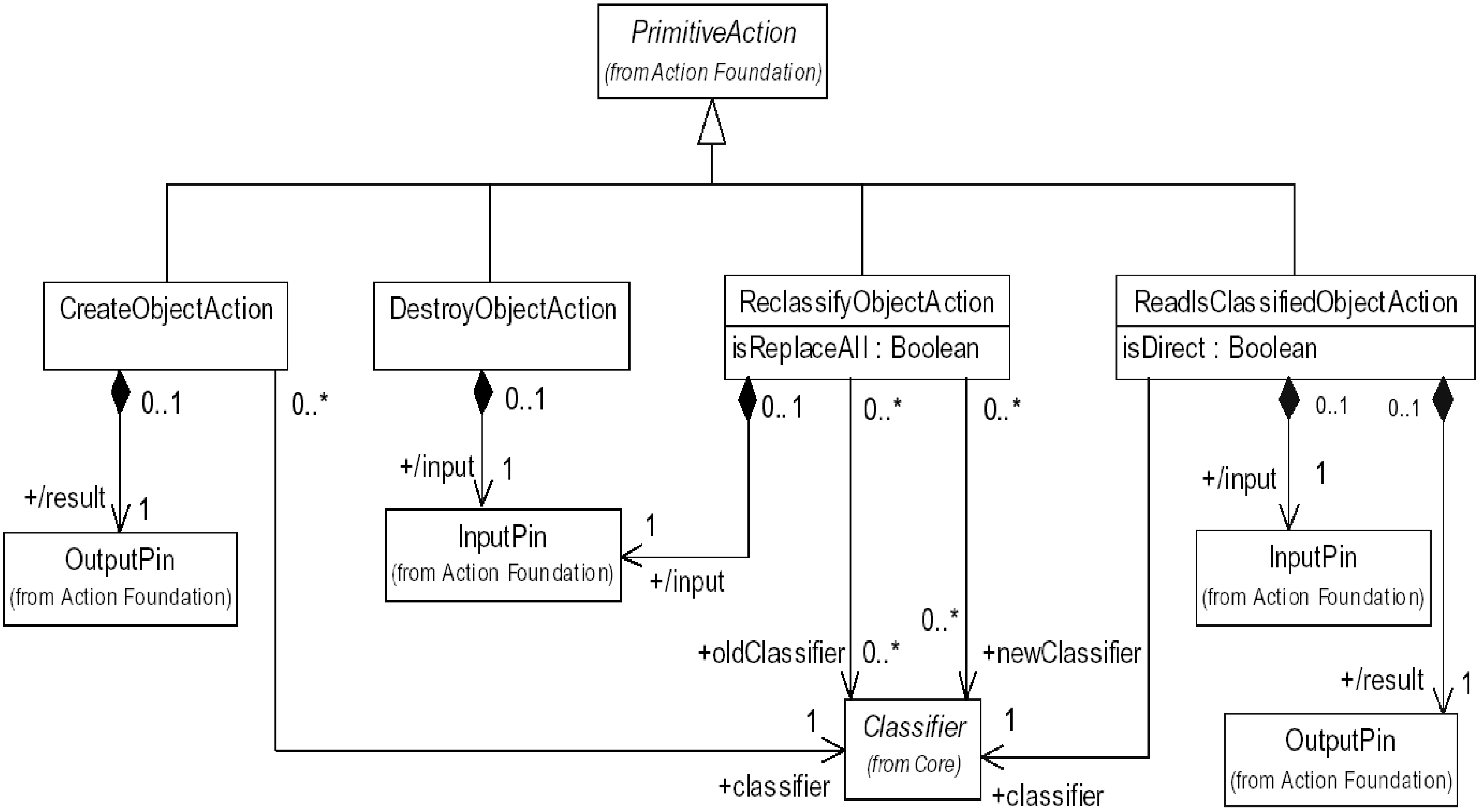
- Actions for synchronous, asynchronous invocation

Jumps

- break, continue, exceptions

Surface languages may define their own actions

UML: Object Action Metamodel



Part 5

Object Action Language, Project Technology

OAL: Object and attribute actions

Create object

- create object instance <objref> of <class>
- ReclassifyObjectAction

Write attribute

- <objref>.<attribute name> = <expression>

Read attribute

- use of ... <objref>.<attribute name> in expressions

Delete object

- delete object instance <objref>

OAL: Link actions

Links are maintained via the relate and unrelate constructs

Create link

- // Create and relate a new b to the given a.
  create object instance b of B;
  relate b to a across R1;

Delete link

- unrelate <source instance handle> from <destination instance handle> across <relationship specification>;
- unrelate <source instance handle> from <destination instance handle> across <relationship specification> using <associative instance handle>;

OAL: Selection expressions

Class extent

- select many <objrefset> from instances of <class>;

Qualification i.e. a single object

- select any <objrefset> from instances of <class> where <where clause>;
  - select any dog related by owner->D[R2] where (selected.name == "Fido");

Qualification i.e. many objects

- select many <objrefset> from instances of <class> where <where clause>;
  - select many dogs related by owner->D[R2] where selected.color == "black";

OAL: Synchronising Objects and Timing

There is no global synchronisation or global time concept in executable UML

Time is local to each concurrently executing object

Einstein´s relativistic view of time

Model Compiler issues:

- The model compiler is required to preserve the explicit synchronisation built into your executable models i.e. deliver each and every signal originating from producers and directed towards consumers

Do not depend on the order of received signals, order is non-deterministic

OAL: Create events and generate signals

generate <signal> to <instance handle>

- select one b1 related by self->L_BU[R1];
  generate L_BU1:ev_toggle to b1;
- create event instance toggle of L_SW1:ev_toggle() to s1;
- my_timer=TIM::timer_start_recurring(microseconds:500000,
  event_inst:toggle);

Beware: an object can be in a single state at a time:

- UML transitions must run to completion
- make them really atomic & instant
- there is no way to limit action activity within transition processing
- It is up to you and your know-how

Part 6

Pathfinder Solutions Action Language - PAL

PAL: data types and basics

- **Boolean, Character, String, Real, Integer,**
- **Constant Declaration**
- **Local Variable Declaration**
- **Assignment action**
- **GenericValue: stores a String, Real, Handle, or Integer (similar to C union)**
- **Handle: generic reference (similar to void* in C)**
- **Group<base_type>, GroupIter<base_type>,**
- **Ref<*class_name*>,**
- **UserDefined enumeration,**
- **UserDefined typedef,**
- **ServiceHandle: allows a run-time dynamic binding mapping, a kind of DII mechanism**

PAL: Conditional, Iteration, Jumps

Conditional:

- **IF (**Boolean Expression**) {** StatementBlock **}**
  [**ELSE IF (Boolean Expression) {StatementBlock}**]
  [**ELSE {** StatementBlock **}**]

Iteration:

- **FOREACH** *cursor_variable* **= CLASS** *class name*
  [**WHERE (**Expression**)**]
  { StatementBlock **}**
- **FOREACH** *cursor_variable* **=** Navigation [**WHERE (**Expression**)**]
  { StatementBlock **}**
- **WHILE (**Expression**) {** StatementBlock **}**

Jumps:

- **BREAK, CONTINUE, RETURN** [Expression]

PAL: Creation, Deletion, Find, Linking

Object creation, deletion

- **CREATE** *class_name*
  [**(** *attribute_name* **=** Expression**,** … **)**] [**IN** *initial_state*]
- **DELETE** *instance_ref*

Finding objects:

- **FIND** [{ **FIRST** | **LAST** }] **CLASS** *class_name* [**WHERE** **(**Expression**)**]

Linking:

- **LINK** *instance1_ref* **A**<*number*> *instance2_ref*
  [**ASSOCIATIVE** *assoc_ref*]
- **UNLINK** *instance1_ref* **A***number instance2_ref*

PAL: Navigation, Event generation

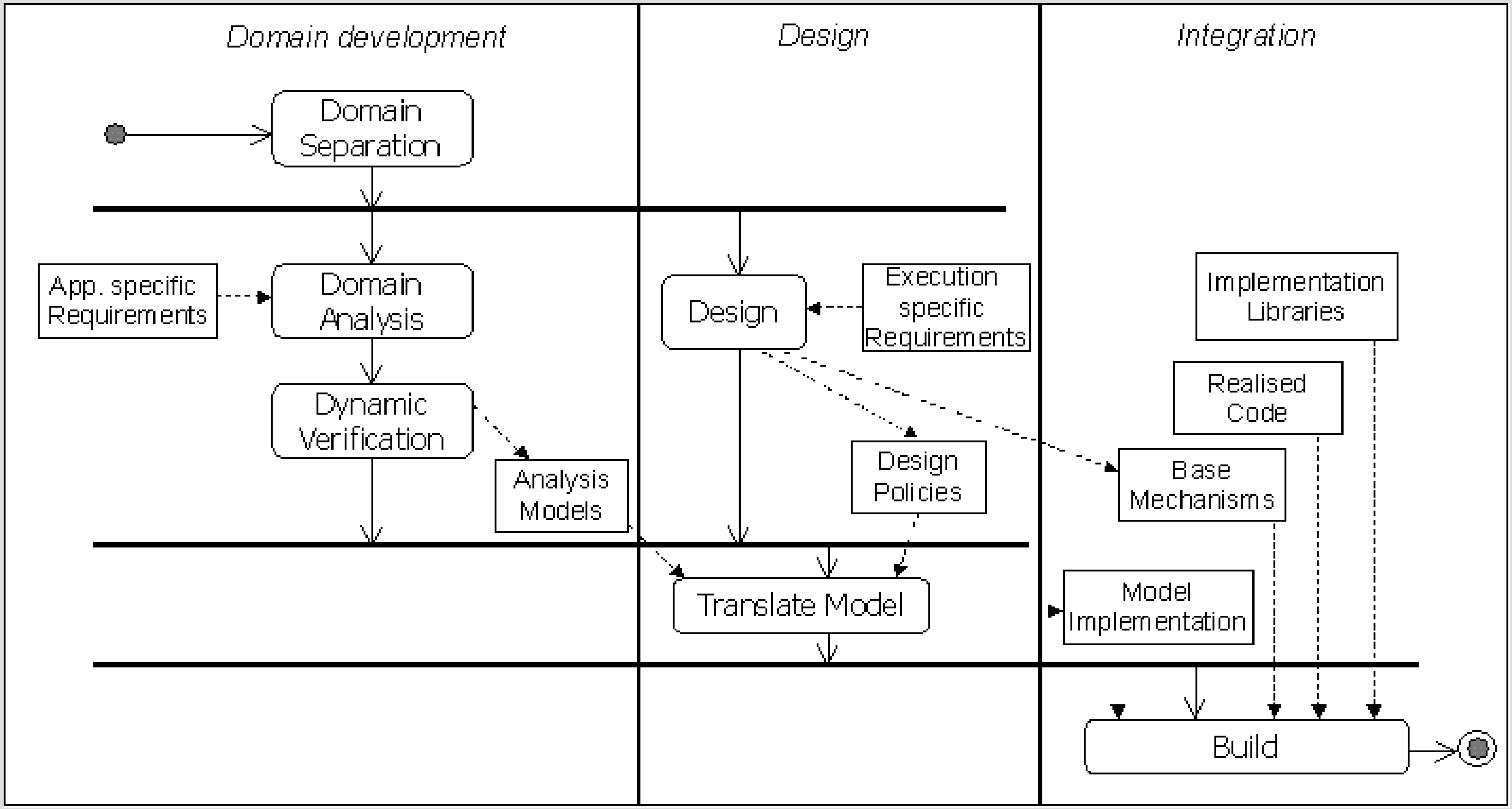
SubSuper Navigation – “downcast” to get from a supertype to a specific subtype. Upcasting is performed automatically. A subtype can be used anywhere a supertype is expected.

- **supertype_ref ->Srelationship_number->subclass_name**

GENERATE ***event_name*** **[** AFTER (***delay***) **]** **[** TO (***destination_ref***) **]**

zühlke

A process for executable UML: MBSE



www.zuehlke.com

Part 7

Demo: Lightland Example

Part 8

Implications and summary

Implications

Separation of model engineering from platform specific software engineering

Creativity is focused on:

- Producing domain models

- Translating models to code

Complete and executable models are produced by domain experts in form of instrumented software

Design i.e. platform specific models are delivered by software engineering teams

Less hindrance with implementation means more time devoted to analysis for domain experts

Summary

A good attempt at solving the software crisis

- perhaps the most far reaching one, until now

Results in more powerful models and our ability to conquer more complexity than ever before.

Brings a new sign of ripeness to the discipline of software engineering

Be prepared for tomorrow‘s challenges

References

UML 1.5 with Action Semantics, Sept 2002

Model Driven Architecture, OMG documentation set

Executable UML, Mellor/Balcer, Project Technology

Pathfinder Solutions, PAL documentation set

Kennedy Carter, iUML documentation set

Convergent Architecture, Richard Hubert

Leon Starr, Executable UML: How to Build Class Models

MDA Course, Jim Arlow, Zuehlke Engineering AG

Part 9

Q/A