OCL 2019 Keynote Retrospective and Prospective

Edward Willink

Willink Transformations Ltd
Eclipse Foundation
MMT Component co-Lead
OCL Project Lead
QVTd Project Lead
QVTo Committer
OMG (Model Driven Solutions)
OCL 2.3, 2.4, 2.5 RTF Chair
QVT 1.2, 1.3, 1.4 RTF Chair

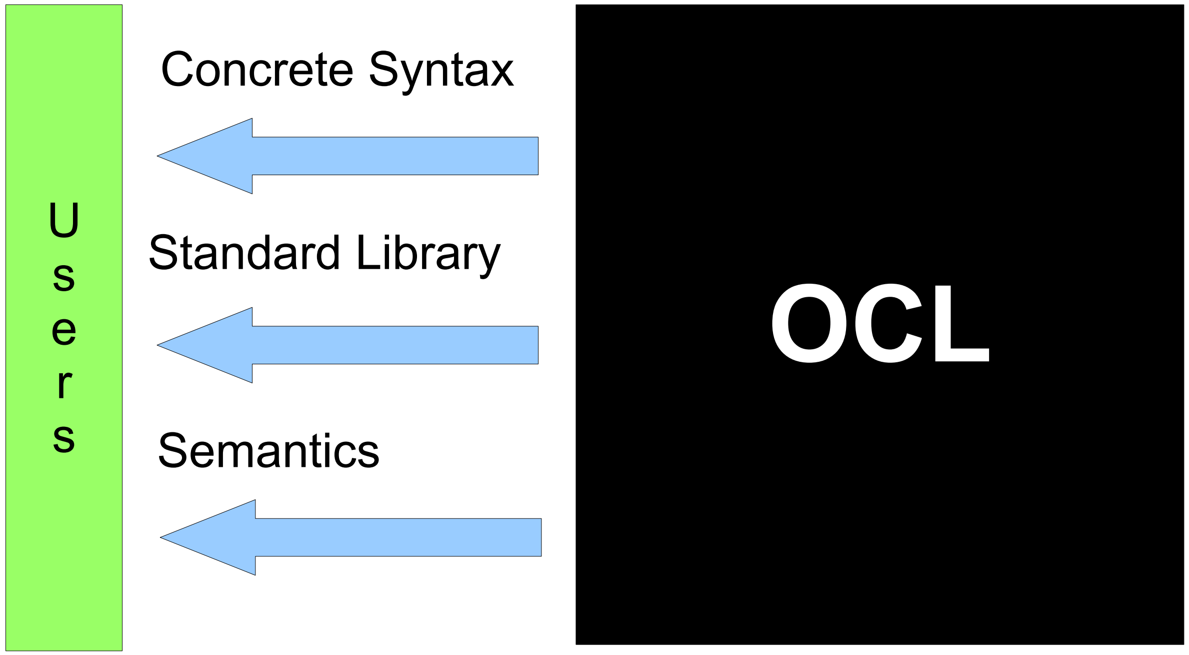
OCL 2019 @ MODELS 2019

16th September 2019

Overview

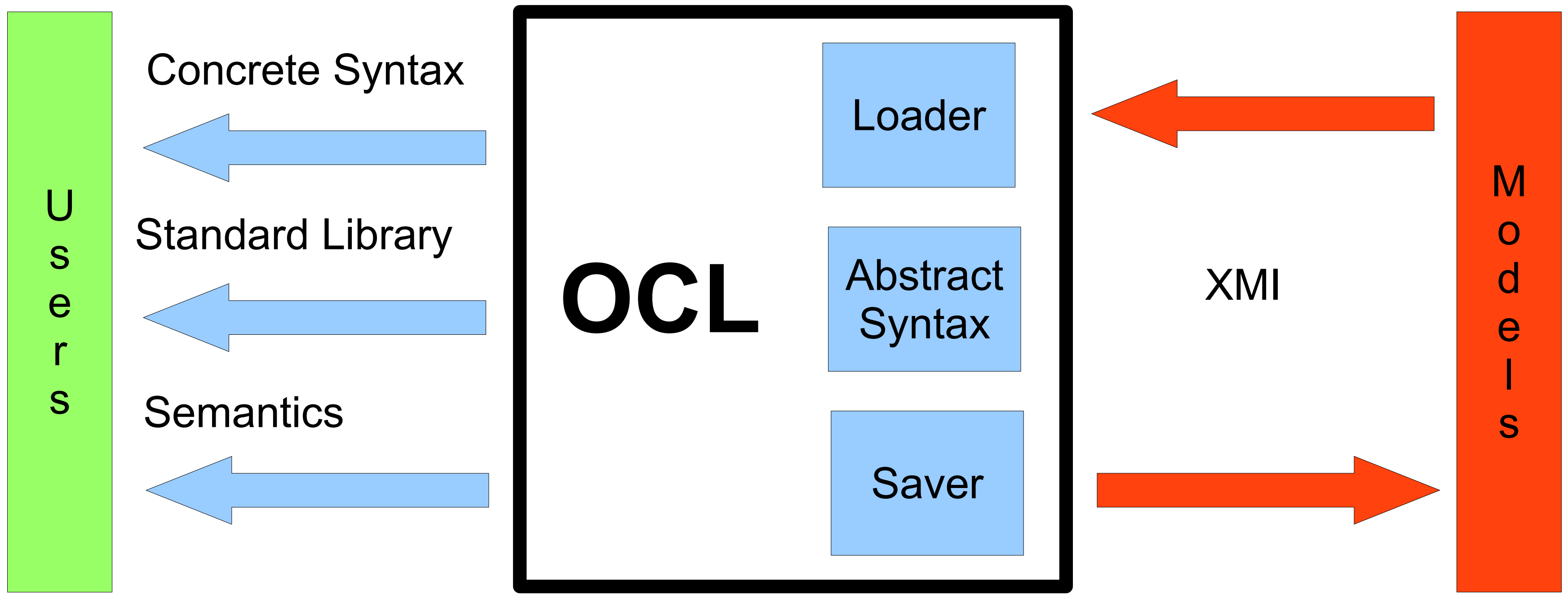
- OCL History
  - OMG process
  - Where is OCL 2.5?
- OCL Tools / USE / Eclipse OCL History
- My involvement / vision
- OCL Problems solved
- Upbeat OCL Prospective

Simple Black Box view of OCL



- OCL specifies computations on model elements
- Specification exposition imperfect
  - Precise Concrete Syntax Grammar
- More functionality

White Box view of OCL



- Specification totally inadequate
  - Missing / Unimplementable
- Rewrite / Design

OCL 1.x History

- Early 1990s, Steve Cook + John Daniels @ IBM
  - Syntropy: OMT diagrams + Z formality
- 1995, Steve Cook + Jos Warmer @ IBM
  - OCL - compromise between Z / programming
- 1997... OCL 1.x bundled as part of UML 1.x
  - constraints only
- 1998 Jos Warmer + Anneke Kleppe
  - The Object Constraint Language: Precise Modeling with UML

OCL 2.0 History

- 2003-2005 UML 2U/U2P competition
  - draft OCL 2.0 ejected as generally useful
    - expressions too, let, minor syntax evolution
- 2003 Jos Warmer + Anneke Kleppe
  - The Object Constraint Language: Second Edition Getting Your Models Ready for MDA
- 2004-2006 8-way QVT competition
  - 7 submissions re-use OCL
- 2006 OCL 2.0 (still draft) published
- 2007 QVT 1.0 published

OCL 2.x

- 2010 OCL 2.1 .. 2.2 - minor fixes
- 2010 OCL 2.3 - minor fixes
  - null/invalid clarification
- 2012 OCL 2.3.1 ISO standard
  - complete with TBDs
- 2014 OCL 2.4 - minor fixes
- 2015 OCL 2.5 Request For Proposal
  - bug list

OMG Process

- Specifications
  - produced by a Finalization Task Force
  - revised by a Revision Task Force
- Specification Problems
  - raised by anyone as OMG issues (now JIRA)
- Task Force membership
  - members represent interested OMG members
    - companies / institutes / ...
  - sometimes paid by the member company

Specification Process

- RTF produces a revised specification
  - Adobe FrameMaker preferred
- RTF produces a report
  - one ballotted resolution per issue
  - resolution responds to an issue (may defer)
    - issue (tided up original text)
    - discussion (condensed email discussions)
    - response (detailed editing instructions)
  - very manual, mixture of source text formats
  - 'Word' preferred

The Response problem

- Trivial one word/paragraph change
  - tedious but ok add/replace instructions
- Changes often impact in multiple places
  - really tedious but ok add/replace instructions
  - often only 95% hit-rate
- Non-trivial changes often overlap
  - really really tedious edit detail sequencing
  - or fix 1 delegates all edits to fix 2
- Non-trivial changes are a mega-pain (unpaid)

The OCL Response Problem

- OCL 2.0 was actually still a draft
  - many TBDs for when UML 2.0 exists
  - numerous pervasive overlapping fixes
    - e.g. UML 2.0 changed "Attribute" to "Property"
- Not enough time/enthusiasm to detail all 'typos'
  - replacement chapter-sized fixes not acceptable
- OCL specification is referential
  - full of references to model concepts
  - auto-generate from models plus comments
    - cf Eclipse OCL Standard Library help pages
    - UML 2.5 is 50% auto-generated

OCL Specification way forward

- OCL 2.5 Request for Proposal
  - Rewrite rather delta edit
  - similar to UML 2.5 rewrite
  - from OMG: (unpaid) solutions from interested parties
- Need to be coherent with other OMG standards
  - fUML introduces stronger semantics for UML
  - similar approach needed for OCL
    - massive (unpaid) research activity
- OCL 2.next Request for Comments
  - from interested party to OMG
    - take it => OMG OCL 2.next specification
    - leave it => Eclipse OCL 2.next specification

OCL specification progress

- Many of the problems solved
  - prototyped in Eclipse OCL
  - extensively modelled for the models
    - needs a clean-up pass
    - remove Eclipse-isms
  - inadequately modelled for the conversions
    - e.g. Normalization of UML stereotypes
    - needs QVTr
- Auto-generation
  - currently models + comments to textile using Xtend
  - probably models + comments to latex using QVTr

Tools

- Oldest - still active
  - IBM/EMF/MDT/Eclipse OCL
  - USE
- Others - not active
  - Dresden OCL
  - Kent OCL
  - Octopus
  - Together
  - ...

UML-based Specification Environment

- Independent self-contained modeling environment
  - variety of UML-related diagrams (integrated)
  - supports OCL evaluation history (Filmstrips)
  - platform for many research papers
  - mainly OCL 1.x, some OCL 2.x (not Ecore)
- 2002 "Development of UML Descriptions with USE"
- 2009 USE 2.4.0 on SourceForge
- 2019-04 USE 5.1.0 on SourceForge

IBM OCL ... EMF OCL ... MDT OCL

- 1995 OCL invented at IBM
- 2000 OMG OCL 2.0 Request for Proposals
- 2001 Eclipse Project seeded by IBM
- 2002 Earliest copyright in Eclipse OCL sources
- 2003 OMG OCL 2.0 draft
- 2004 Eclipse Foundation established
- 2005-09-05 First EMF OCL Bugzilla
  2005-10-27 Migration of org.eclipse.emf.ocl plugins
- <= 2006-06 (1.0.0) (Classic) OCL for Ecore
- 2007-06 (1.1.0) (Classic) OCL for UML 2.1

```
Environment<PK, C, O, P, EL, PM, S, COA, SSA, CT, CLS, E>
```

- 2008-06 (1.2.0) Validation implemented
  - OCL Tools for OCL UI - never progressed

OCL tooling approach

- Claas Wilke, Michael Thiele, Christian Wende
  Extending Variability for OCL Interpretation, OCL 2010
  - User / 'OCL', metaclasses / instances
- Dresden OCL
  - adapt User metaclasses to 'OCL' metaclasses
  - adapt User instances to 'OCL' instances
- Eclipse Ecore / UML OCL
  - re-tool OCL for Ecore / UML metaclasses
  - re-tool OCL for Ecore / UML instances
- Eclipse Pivot OCL
  - normalize User metaclasses as 'OCL' metaclasses (tens)
  - accommodate Ecore / UML instances (thousands)

MDT OCL ... Eclipse OCL

- 2010-06 (3.0.0) EMF delegates
  - Complete OCL / OCLinEcore / OCLstdlib Editors
  - initially LPG2+IMP, then Xtext
- 2011-06 (3.1.0) Pivot OCL as 'examples'
  - Impact Analyzer (deprecated) - Classic Ecore OCL
- 2012-06 (4.0.0) Java Code Generator
- 2014-06 (5.0.0) Debugger, Validity View
- 2015-06 (6.0.0) Pivot OCL not 'examples'
- 2018-06 (6.4.0) EMF EAnnotation validators
  - builder + nature
- 2019-09 (6.9.0) Latest 3-monthly Eclipse release

Eclipse OCL - not implemented

- UML States - oclIsInState()
  - parsed but otherwise ignored
    - EMF objects do not have state, need UML objects
- UML Messages - ^, ^^, hasChanged(), hasSent()
  - parsed but otherwise ignored
    - needs history
- OCL Evaluation history - oclIsNew(), @pre, pre, post
  - parsed but otherwise ignored
    - major design compromise wrt memory / speed
      - https://bugs.eclipse.org/538468
    - instrumenting EMF construction is costly
    - **pre** could be a code generation option
- Use USE for history support

Solutions

- OCL 2.5 RFP identifies many problems
- Eclipse Pivot OCL
  - free from legacy Ecore/UML OCL compatibility
  - free to prototype solutions

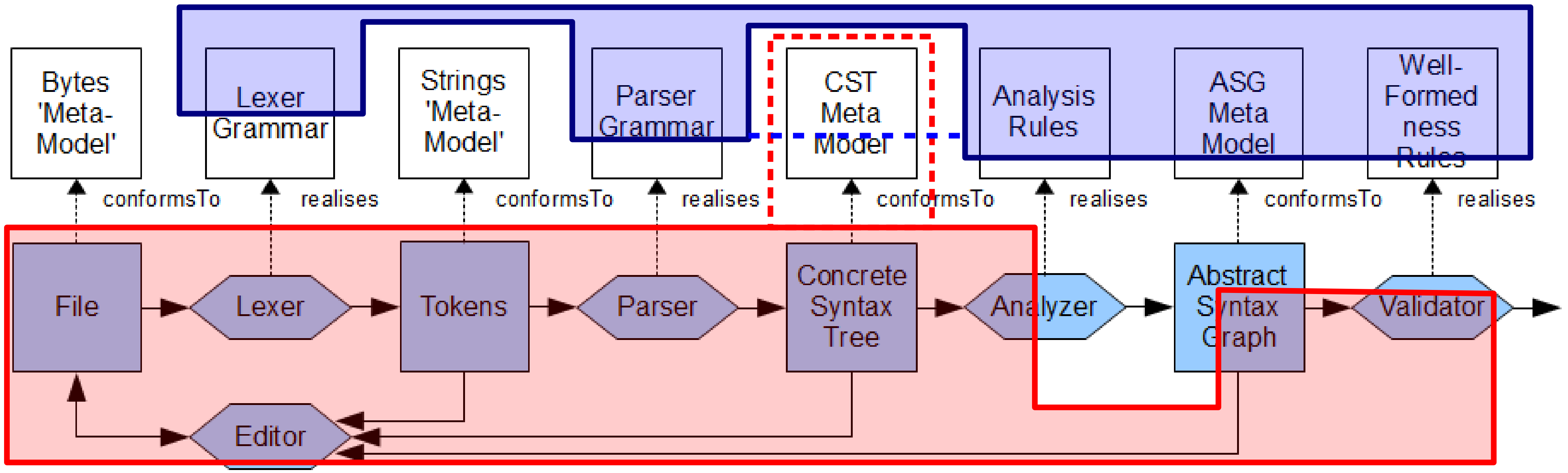
Pivot OCL solutions 1

- Models
  - OMG OCL has a broken Abstract Syntax model
  - Pivot OCL provides extensible (QVTr extends)

Abstract Syntax model	UML -> Ecore -> Java
Well Formedness Rule model	Complete OCL -> Ecore -> Java
Concrete Syntax models	Ecore -> Java
Grammars	Xtext -> Java
Standard Library Model	OCLstdlib -> Java
Code Generator Model	Ecore -> Java

  - Pivot OCL substantially auto-generated from its models
    - many use transformation from UML source

Specification-to-compiler Automation



OCL specification is incomplete

Xtext covers large parts of an implementation

Pivot OCL solutions 2

aValue.aFunction()

- Overloading / dynamic dispatch
  - OMG OCL uses UML operations
  - UML operations have implementation(OCL)-defined semantics
  - Pivot OCL (and Classic OCL) provides Java-like overloading

Pivot OCL solutions 3

anObject.oclType().name

- Reflection
  - OMG OCL excludes MOF reflection
  - OMG OCL specification uses oclType() reflectively
  - OMG OCL oclType() is query / powerset / open enumeration
  - Pivot OCL oclType() returns a reflective OCL metaclass

Pivot OCL solutions 4

Complex{real=1.0,imag=0.0}

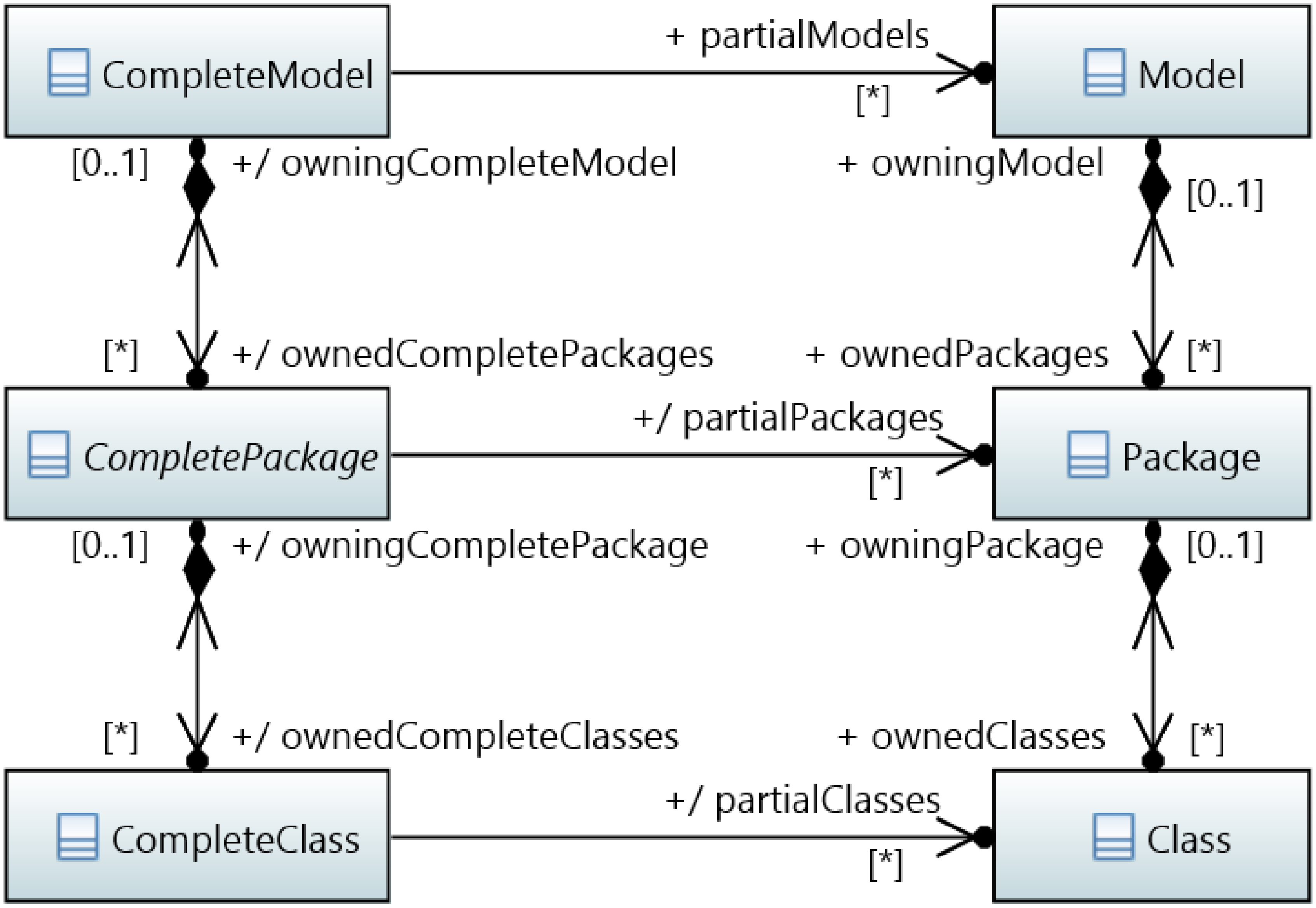
- Type construction
  - OMG OCL has no type construction
  - Pivot OCL has Shadow objects

Pivot OCL solutions 5

context Element
def: isPersistent() : Boolean[1] = false

- Open Classes
  - UML has Model, Package, Class, ...
    - no additions => Closed Classes
  - OMG OCL has magic Complete OCL
    - additional features that behave as ordinary features
      => Open Classes
  - Pivot OCL - CompleteModel, CompletePackage, CompleteClass
    - each is an overlay of UML Model, Package, Class
    - modelled, extensible, has URI

OCL Complete Model



Pivot OCL solutions 6

- XMI
  - OMG requires AS serialization - many problems
  - Pivot OCL AS is serializable
    - Complete classes for Complete OCL open classes
    - Orphanage package for synthetic types
    - Blindness to Primitive/Collection/Tuple type namespaces
    - OCL standard library has a referenceable model

Pivot OCL solutions 7

Sequence(String)

- Templates
  - UML has generic types / templates
  - OMG OCL has e.g. Sequence(T) with magic T
  - Pivot OCL has templated classes / operations
    - UML's TemplateParameterSubstitution etc
    - Sequence(T) - T is a template parameter

Pivot OCL solutions 8

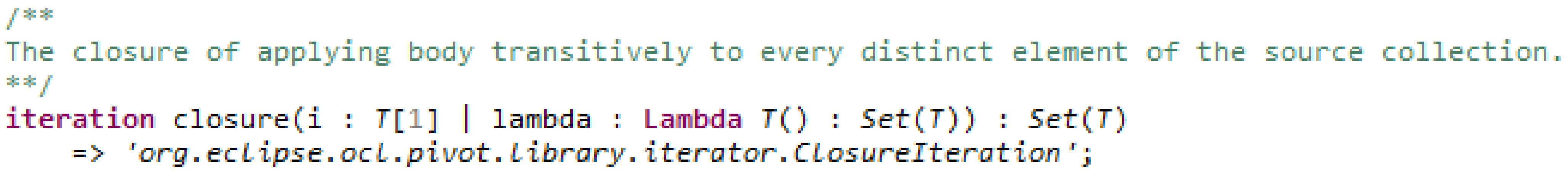
aCollection->select(e | **e.name.size() < 5**)

- Lambdas
  - OMG OCL Iterator body is obviously a lambda expression
    - closure() specification uses text macro exposition

      an invocation-site-specific helper function
      synthesized by lexical substitution of
  - Pivot OCL uses lambdas, types and variables

Pivot OCL solutions 9

- Library Model
  - OMG OCL is textual
  - Pivot OCL has standard library model (and editor)



```
/**
The closure of applying body transitively to every distinct element of the source collection.
**/
iteration closure(i : T[1] | lambda : Lambda T() : Set(T)) : Set(T)
    => 'org.eclipse.ocl.pivot.library.iterator.ClosureIteration';
```

Pivot OCL solutions 10

anObject.name?.size()

- Null safety
  - UML has optional multiplicity T[?]
  - Pivot OCL support [?] / [1] multiplicities nullable/non-null
  - Pivot OCL adds ?. and ?-> null safe navigation

Pivot OCL solutions 11

- Map(K, V)
  - see lightning talk

Pivot OCL solutions 12

- Stereotypes
  - UML has Stereotypes
  - UML has a base_xxx, extension_xxx navigation indication
  - OMG OCL ignores Stereotypes
  - Classic OCL relies on Eclipse UML2 Java API
  - Pivot OCL normalizes as regular classes / navigation
- Association Classes
  - UML has Association Classes
  - OMG OCL has hard to understand special semantics
  - Pivot OCL normalizes as regular classes / navigation

Pivot OCL not yet solutions 1

- Lazy collections
  - lazy can dramatically improve partial evaluation
  - lazy is an overhead for full evaluations
- Deterministic collections
  - OCL 'should' be deterministic
  - not that hard / inefficient
- Patterns
  - solves oclIsKindOf/oclAsType redundancies ...
    if x.oclIsKindOf(Z) then x.oclAsType(Z).f() else null endif
  - must use QVTr syntax
  - grammar is challenging - operation/iteration

Pivot OCL not yet solutions 2

- Modules
  - auto-tooling can support pick and mix
    - discard States, discard Messages, add Temporal
  - blocked by modular grammar tooling
- Evaluation semantics
  - auto-tooling would be nice
  - awaits academic input

System Software Vision

- 1978-2012 Embedded System Engineer
  - Digital Signal Processing - DSP
  - disappointing tools - do better
- Vision - reliable systems
  - system is composed of sub-systems
    - need a solid composition semantics
      - Waveform Description Language (WDL)
    - need a solid leaf sub-system semantics
      - CAL Action Language

System Engineering

- System and sub-systems
  - Traditionally sub-systems (components) are re-used
    - system has to accommodate sub-system eccentricities
  - Sub-systems should satisfy the needs of the system
    - sub-system should be tailored (auto-generated)
- System concerns
  - scheduling, deployment, precision, ...
    - each can be analyzed / auto-generated
- Auto-generation
  - gcc should be a 100 transformation chain
    - a PhD could research/rewrite just one

DSP-specific Tooling

- 1996-2001 PhD - DSP compiler optimizations
  - meta-compilation for C++
- 2001 Reflection - Two worlds
  - DSP world (small community)
    - interesting concurrency ideas Ptolemy@UCB
    - poor tools - inverse CASE law
  - Java world (large community)
    - good tools
    - ad hoc / poor concurrency

Early Transformation Activities

- CAL @ UCB - an XSLT transformation cascade
  - Seriously unreadable => NiceXSL
- CAL - yet another small community language
  - not significantly different to a wrapper around OCL

Eclipse GMT Activities

- 2004 Eclipse GMT (Jean Bézivin)
  - UMLX 1 inspired by OOPSLA GME paper
    - Outline NiceXSL transformation design
  - UMLX 2 based on GEF
    - No transformation to executble
  - UMLX 3 based on GMF
    - No transformation to executble

Early QVT / OCL activities

- UMLX '4' do execution first, graphics last
  - UMLX and QVTr rather similar
    - alternate graphical/text views
  - Develop/exploit QVTr first
    - UMLX's QVT editors contributed to QVTd
    - OCL extensibility contributed to MDT OCL
  - [2017 UMLX '5' using Sirius+QVTr+OCL+Java]
- 2008 Eclipse QVTd committer
- 2009 Eclipse OCL committer

E.D.Willink Roles/Activities 2

- 2009 Eclipse OCL project lead
- 2010 Eclipse QVTd project lead
- 2012 ----- 'retired' -----
- 2012 Eclipse QVTo committer
- 2012 OMG OCL RTF chair
- 2013 OMG QVT RTF chair
- 2014 OCL 2.4 - minor tidy up/clarification
- 2014 OCL 2.5 RFP - bug list
- 2015 QVT 1.2 - minor tidy up/clarification
- 2016 QVT 1.3 - minor tidy up/major clarification

OCL State of the Art

- OCL developer base is probably declining
  - no commercial tools - all failed / Open Source
  - many tool failures - Dresden OCL most recently
  - two 'ok' quality implementations
  - OCL has less than ten developers
  - OCL specification is poor
  - OCL specification progress is pitiful
- OCL user base is perhaps steady
  - many modeling papers use OCL
    - little alternative - OCL is 'right'

Updated Reflection

- 2001 Reflection - Two worlds
  - DSP world (small community)
  - Java world (large community)
- 2019 Reflection - Two worlds
  - OCL world (tiny community)
  - Java world (huge community)

Java contrast

- Java etc keeps improving (similar good ideas)
  - generics, lambdas, streams, ...
  - Java has multiple high quality implementations
    - some free
  - Java has thousands of developers, OCL barely ten
  - @Pure is not practical

- OCL cannot outdo Java on Java's ground
  - find another playing field

Language Progress as Bye-Byes

- Assembler
  - bye bye 1's and 0's
- C
  - bye bye stack/condition code corruption
  - bye bye self-modifying code
- Object Orientation
  - nearly bye bye global variable corruption
  - nearly bye bye distant variable corruption
- Java
  - bye bye heap corruption
- OCL
  - bye bye side effects
  - bye bye memory corruption

Why OCL?

- OCL has just one Unique Selling Point left
  - side effect free
    - analyzeable, predictable, reliable, (fast)
- But OCL is useless
  - side effect free prohibits mutation - no results
  - needs embedding in a model provider
    - OCLinEcore helps
    - Model Transformation really helps

Incremental (bidirectional) Declarative Transformation

- Underlying OCL
  - rigorous queries / analyzeable memory accesses
- Declarative Transformation
  - limited / analyzeable mutations
  - global dependency analysis
  - reliable computation scheduling
  - reliable selective / incremental re-computation
- Manual Java cannot sensibly do this
  => OCL / QVTr Killer Application

OCL for proofs

- Not my field - just my prejudice
  - need to challenge a specification
    - is it sensible
    - is a behaviour guaranteed / impossible
- cf. Declarative Model Transformation
  - metamodel type system very restrictive
  - eliminates numerous scheduling alternatives
  - potentially NP complete but O(1) or O(2) in practice
- Contradiction finders
  - transform to existing non-metamodel tools
    - really really slow for moderate problems
  - ?? remain metamodel aware and exploit OCL ??

Summary

- OCL could have an important future
- drastic tool improvements needed
  - quality
  - ease-of-use / fun-to-use
  - documentation / examples
- Fast QVTr incremental tooling is happening
- OCL-based proof tools needed
  - pre/post/frame condition / history unification - USE
- ?? Questions ??