# *XMF and XModeler*

Tony Clark[1]

[1]School of Engineering and Information Sciences
University of Middlesex

August 6, 2011

## Outline

## *OMG, UML 2.0*

- Around 1999 Clark, Evans, Kent started attending OMG.
- UML 2.0 started around this time.
- The 2U Submission: UML as family of languages:
  - Templates.
  - Package Extension.
- Tools and methods emerged for model-based language engineering:
  - Language Architectures.
  - Denotational Semantics.
  - Operational Semantics.

## *Modelling and Programming*

- Aim: to merge modelling and programming.
- Tools: MMT, XMT, XMF.
- Programming language based on FP and OCL.
- Important features: meta-; reflection; OO.
- Tools for Language Engineering.

## *Xactium*

- Clark, Evans set up in 2003.
- Developed XModeler (XMF-Mosaic) on top of XMF.
- 2003-2008.
- Clients: BAES; BT; Citi-Group; Artisan; BSkyB.

## XMF

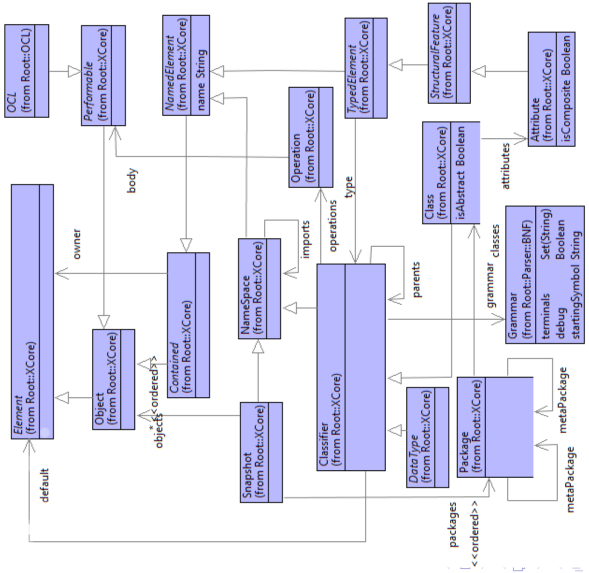
- Meta-Circular Language (like MOF and ECore).
- XCore Based on ObjvLisp.
- File based or world-state.
- Features for:
    - Packages of models/programs.
    - Higher-order operations.
    - OCL.
    - Meta-Object Prototcol (MOP).
    - Language Engineering (grammars, syntax processing).
    - Daemons (object listeners).
    - Pattern matching.
    - Code generation templates.
    - Threads.
    - XML processing (parsing).
    - Java integration.

## *XModeler*

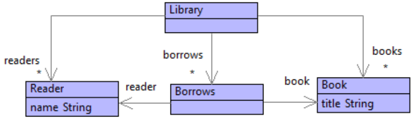
- Eclipse RCP Tool.
- Layered on (and written in) XMF.
- MVC Architecture.
- File interface for building XMF applications.
- Functional interface e.g. deploy XML, check constraints.
- Clients are all extensible:

  *Client:* XMF Command Listener.
  *Client:* Browsing all data.
  *Client:* Editing all data.
  *Client:* Diagrams: class; snapshot.
  *Client:* Text editing, HTML browsing.

## Meta-language

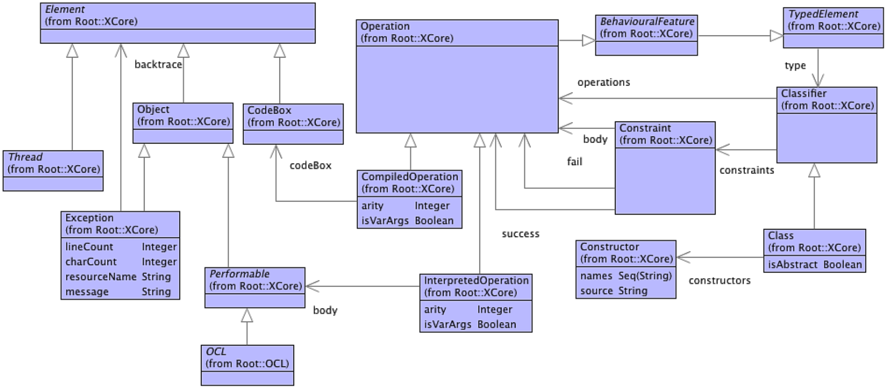

OCL (from Root::OCL)
Performable (from Root::XCore)
NamedElement (from Root::XCore) name String
TypedElement (from Root::XCore)
StructuralFeature (from Root::XCore)
Attribute (from Root::XCore) isComposite Boolean
Operation (from Root::XCore)
Class (from Root::XCore) isAbstract Boolean
attributes
body
type
owner
Element (from Root::XCore)
Contained (from Root::XCore)
NameSpace (from Root::XCore)
imports
operations
parents
grammar
classes
Grammar (from Root::Parser::BNF) terminals Set(String) debug Boolean startingSymbol String
Object (from Root::XCore)
<<ordered>>
objects *
Snapshot (from Root::XCore)
Classifier (from Root::XCore)
DataType (from Root::XCore)
Package (from Root::XCore)
metaPackage
metaPackage
default
packages
<<ordered>>

## Models



```
parserImport XOCL;

context Root
  @Package BasicLibrary
    @Class Library
      @Attribute books   : Set(Book)    end
      @Attribute readers : Set(Reader)  end
      @Attribute borrows : Set(Borrows) end
    end
    @Class Book
      @Attribute title : String end
    end
    @Class Reader
      @Attribute name : String end
    end
    @Class Borrows
      @Attribute reader : Reader end
      @Attribute book : Book end
    end
  end
```

## *Behaviour*



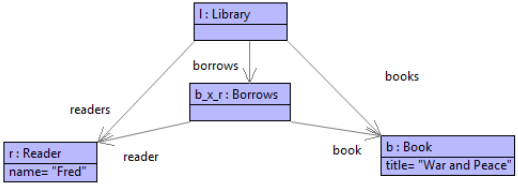
Element (from Root::XCore)
Thread (from Root::XCore)
Object (from Root::XCore)
CodeBox (from Root::XCore)
backtrace
codeBox
Exception (from Root::XCore)
lineCount Integer
charCount Integer
resourceName String
message String
Performable (from Root::XCore)
OCL (from Root::OCL)
body
Operation (from Root::XCore)
CompiledOperation (from Root::XCore)
arity Integer
isVarArgs Boolean
InterpretedOperation (from Root::XCore)
arity Integer
isVarArgs Boolean
success
fail
body
operations
BehaviouralFeature (from Root::XCore)
TypedElement (from Root::XCore)
type
Classifier (from Root::XCore)
Constraint (from Root::XCore)
constraints
Class (from Root::XCore)
isAbstract Boolean
Constructor (from Root::XCore)
names Seq(String)
source String
constructors

## Programs: XOCL

```
parserImport XOCL;
import BasicLibrary;

context Book
  @Constructor(title) end

context Library
  @Operation addBook(title:String)
    let b = Book(title)
    in self.books := books->including(b); b
    end
  end

context Library
  @Operation findBook(title:String):Book
    let B = books->select(b | b.title = title)
    in if B->isEmpty
       then null
       else B->asSeq->head
       end
    end
  end
```

## *Snapshots*

```
context Root
  @Operation test()
    let S = Snapshot("library",Seq{BasicLibrary})
    in S.add("l",Library());
       let l = S::l
       in S.add("b",l.addBook("War and Peace"));
          S.add("r",l.addReader("Fred"));
          S.add("b_x_r",l.addBorrows(S::r,S::b));
          S
       end
    end
  end
```



## Defining Constraints (1)

```
context Library
  @Constraint all_borrowed_books_belong_to_library
    borrows.book->forAll(b | books->includes(b))
    fail "can only borrow books in library"
  end

context Library
  @Constraint all_borrowing_readers_belong_to_library
    borrows.reader->forAll(r | readers->includes(r))
    fail "only registered readers can borrow books"
  end
```

## *Defining Constraints (2)*

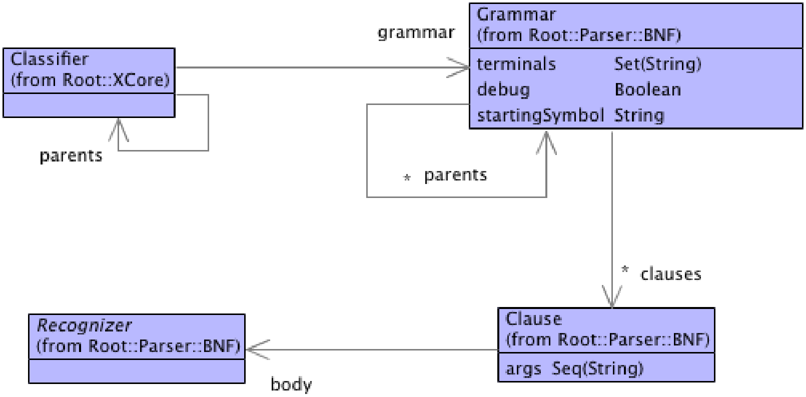
```
context Library
  @Constraint cannot_borrow_same_book_twice
    borrows->forAll(b1 | borrows->forAll(b2 | b1 <> b2 implies b1.book <> b2.book))
    fail "cannot borrow the same book twice"
  end

context Library
  @Constraint all_books_have_unique_titles
    books->forAll(b1 | books->forAll(b2 | b1 <> b2 implies b1.title <> b2.title))
    fail "books should have unique titles"
  end

context Library
  @Constraint all_readers_have_unique_names
    readers->forAll(r1 | readers->forAll(r2 | r1 <> r2 implies r1.name <> r2.name))
    fail "readers should have unique names"
  end
```

## *Checking Constraints*

```
context Root
  @Operation test_illegal()
    let S = Snapshot("library",Seq{BasicLibrary});
        b = Book("War and Peace")
    in S.add("l",Library());
       S.add("r",(S::l).addReader("Fred"));
       S.add("b_x_r",(S::l).addBorrows(S::r,b));
       S
    end
  end
```

```
[1] XMF> test_illegal().checkConstraints().failures();
Seq{ConstraintReport(<Library d9ff5f>,
     <Constraint all_borrowed_books_belong_to_library>,
     false, can only borrow books in library)}
[1] XMF>
```

## Syntax Classes



## Quasi-Quotes

```
context Root
  @Operation add1_exp(exp:Performable):Performable
    [| 1 + <exp> |]
  end

context Root
  @Operation seq_exp(exps:Seq(Performable)):Performable
    exps->iterate(e x = [| Seq{} |] |
      [| <x>->including(<e>) |])
  end
```

```
[1] XMF>  add1_exp([| x + y |]);
BinExp(IntExp(1),+,BinExp(Var(x),+,Var(y)))
[1] XMF> seq_exp(Seq{[| 1 |],[| x |],true.lift()});
CollExp(CollExp(CollExp(SetExp(Seq,Seq{}),including,Seq{IntExp(1)}),including,Seq{Var
      (x)}),including,Seq{BoolExp(true)})
[1] XMF> seq_exp(Seq{[| 1 |],[| x |],true.lift()}).pprint();
Seq{}->including(1)->including(x)->including(true)
[1] XMF>
```

## Grammars

```
parserImport Parser::BNF;
parserImport XOCL;

Root::g :=
  @Grammar
    Start ::= i=Int o=Op j=Int {
      @Case o of
        "+" do i + j end
        "*" do i * j end
      end
    }.
    Op ::= '+' { "+" } | '*' { "*" }.
  end;
```

```
[1] XMF> g.parseString("1 + 2","Start",Seq{});
3
[1] XMF>
```

## Embedded Language Features (Usage)

```
context Library
  @Subset all_borrowed_books_belong_to_library
    borrows.book books
  end

context Library
  @Subset all_borrowing_readers_belong_to_library
    borrows.reader readers
  end

context Library
  @Unique cannot_borrow_same_book_twice
    borrows book
  end

context Library
  @Unique all_books_have_unique_titles
    books title
  end

context Library
  @Unique all_readers_have_unique_names
    readers name
  end
```

## *Embedded Language Features (Def 1)*

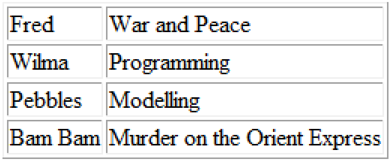
```
parserImport XOCL;
parserImport Parser::BNF;

context Root
 @Class Unique
  @Grammar
   Unique ::= name=Name collection=Name field=Name {
    [| @Constraint unique self.<collection>->forAll(x |
         self.<collection>->forAll (y |
           x <> y implies x.<field> <> y.<field>))
       end.name := "UNIQUE:" + <name.lift()> |]
   }.
  end
 end
```

## Embedded Language Features (Def 2)

```
parserImport XOCL;
parserImport Parser::BNF;
import OCL;

context Root
  @Class Subset
    @Grammar
      Subset ::= name=Name sub=Path super=Path {
        [| @Constraint contained <sub>->forAll(x |
             <super>->includes(x))
           end.name := "CONTAINED:" + <name.lift()> |]
      }.
      Path ::= root=Name fields=('.' Name)* {
        fields->iterate(field exp = Var(root) |
          [| <exp>.<field> |])
      }.
    end
  end
```

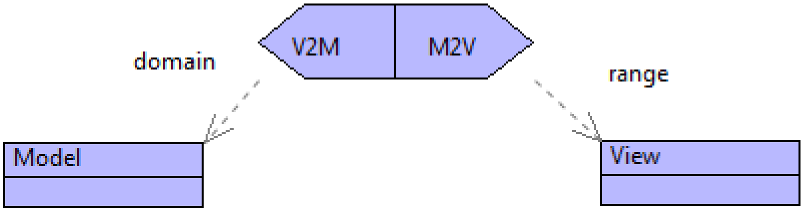
## *Generating Code*

```
context Library
  @Operation borrowsTable()
    let sout = StringOutputChannel()
    in @HTML(sout,0)
         <TABLE border=1>
           { @Loop borrow in borrows do
               [ <TR>
                   <TD> { borrow.reader.name } </TD>
                   <TD> { borrow.book.title } </TD>
                 </TR> ]
             e_nd }
         </TABLE>
       end;
       sout.getString()
    end
  end
```



| | |
|---|---|
| Fred | War and Peace |
| Wilma | Programming |
| Pebbles | Modelling |
| Bam Bam | Murder on the Orient Express |

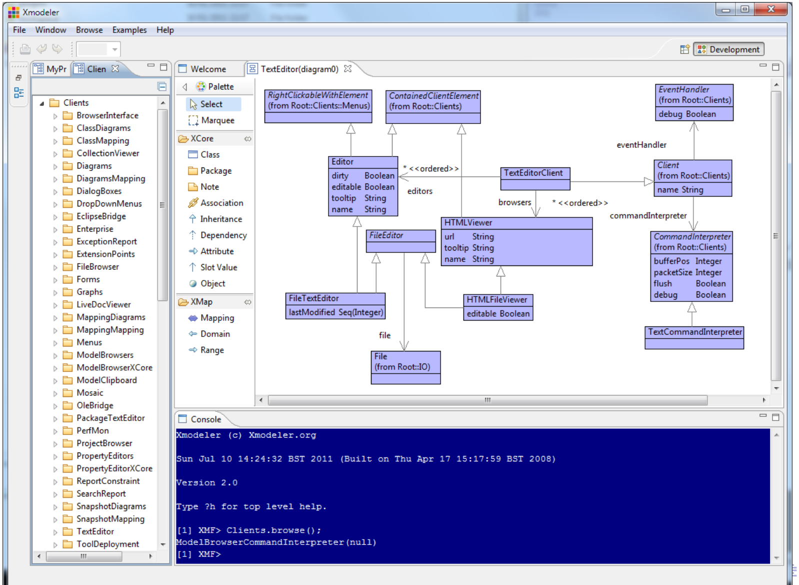
## *Tooling Requirements*



## Adding Daemons

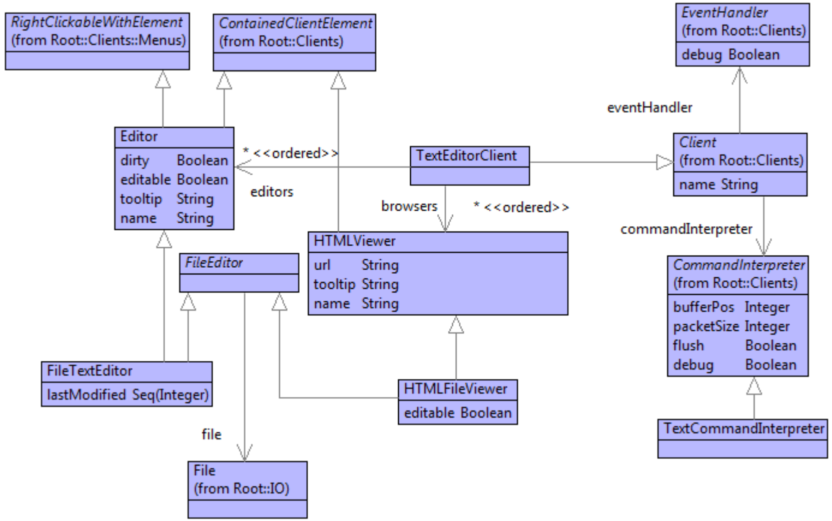
```
o.addDaemon(
  @Operation(slot,new,old)
    ... something to do...
  end)
```

# Clients



```
Xmodeler (c) Xmodeler.org

Sun Jul 10 14:24:32 BST 2011 (Built on Thu Apr 17 15:17:59 BST 2008)

Version 2.0

Type ?h for top level help.

[1] XMF> Clients.browse();
ModelBrowserCommandInterpreter(null)
[1] XMF>
```

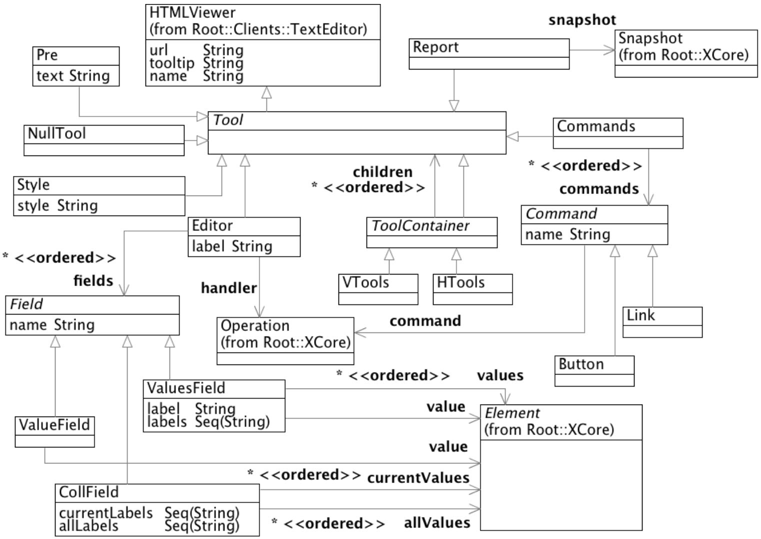
## HTML


RightClickableWithElement (from Root::Clients::Menus)
ContainedClientElement (from Root::Clients)
EventHandler (from Root::Clients)
debug Boolean
eventHandler
Editor
dirty Boolean
editable Boolean
tooltip String
name String
* <<ordered>>
editors
TextEditorClient
Client (from Root::Clients)
name String
browsers
* <<ordered>>
commandInterpreter
FileEditor
HTMLViewer
url String
tooltip String
name String
CommandInterpreter (from Root::Clients)
bufferPos Integer
packetSize Integer
flush Boolean
debug Boolean
FileTextEditor
lastModified Seq(Integer)
HTMLFileViewer
editable Boolean
TextCommandInterpreter
file
File (from Root::IO)

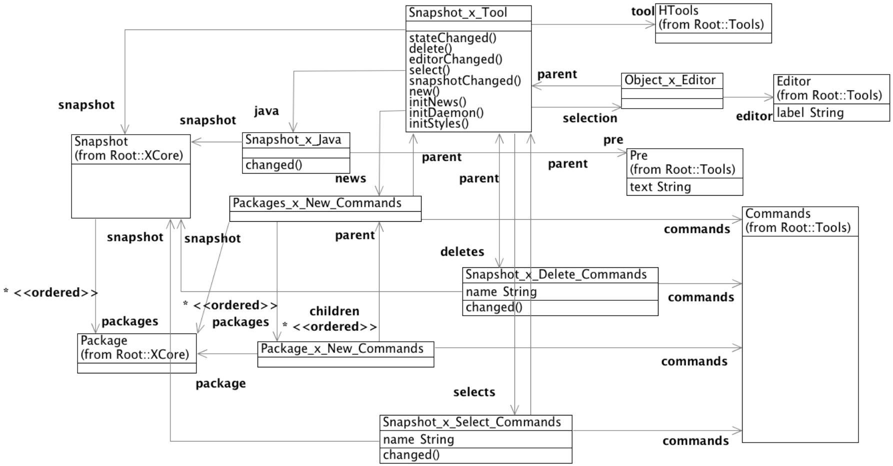
## *Tool Requirements*

- Take any snapshot.
- Interactively construct class instances.
- Interactively edit class instances.
- Check constraints after each modification.
- Generate Java code for the snapshot.

## Tool Models


HTMLViewer
(from Root::Clients::TextEditor)
url String
tooltip String
name String
snapshot
Report
Snapshot
(from Root::XCore)
Pre
text String
NullTool
Tool
Commands
Style
style String
children
* <<ordered>>
* <<ordered>>
commands
Editor
label String
ToolContainer
Command
name String
* <<ordered>>
fields
handler
VTools
HTools
Field
name String
Operation
(from Root::XCore)
command
Link
Button
* <<ordered>>
values
ValuesField
label String
labels Seq(String)
value
Element
(from Root::XCore)
ValueField
value
* <<ordered>>
currentValues
CollField
currentLabels Seq(String)
allLabels Seq(String)
* <<ordered>>
allValues

# MVC Model


Snapshot_x_Tool
stateChanged()
delete()
editorChanged()
select()
snapshotChanged()
new()
initNews()
initDaemon()
initStyles()
tool
HTools
(from Root::Tools)
parent
Object_x_Editor
Editor
(from Root::Tools)
label String
editor
selection
snapshot
snapshot
java
Snapshot
(from Root::XCore)
Snapshot_x_Java
changed()
news
parent
parent
parent
pre
Pre
(from Root::Tools)
text String
Packages_x_New_Commands
Commands
(from Root::Tools)
commands
snapshot
snapshot
parent
deletes
Snapshot_x_Delete_Commands
name String
changed()
commands
* <<ordered>>
* <<ordered>>
children
packages
packages
* <<ordered>>
Package
(from Root::XCore)
Package_x_New_Commands
commands
package
selects
Snapshot_x_Select_Commands
name String
changed()
commands

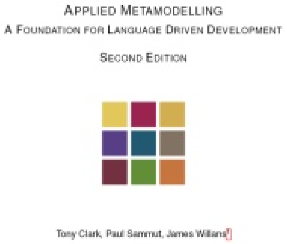
## *Demo*

## *Availability, Documentation, Research*

http://www.eis.mdx.ac.uk/staffpages/tonyclark/

Superlanguages
Developing Languages and Applications with XMF
First Edition

Tony Clark, Paul Sammut, James Willans



Applied Metamodelling
A Foundation for Language Driven Development
Second Edition

Tony Clark, Paul Sammut, James Willans

X M F
The Extensible Programming Language

XPL