Model Evolution: Comparison between Clafer and Textual UML

Presented By: Dina Omar Zayan
Presentation Date: 5/4/2012
CS846

Outline

- Research Problem
- Clafer Model
- Textual UML Model
- Methodology
- Analysis and Evaluation
- Conclusions
- Future Work

Introduction

- What is Domain Modeling? Why is it important?
- Why is Model Evolution important at the requirements stage?

Modeling Languages

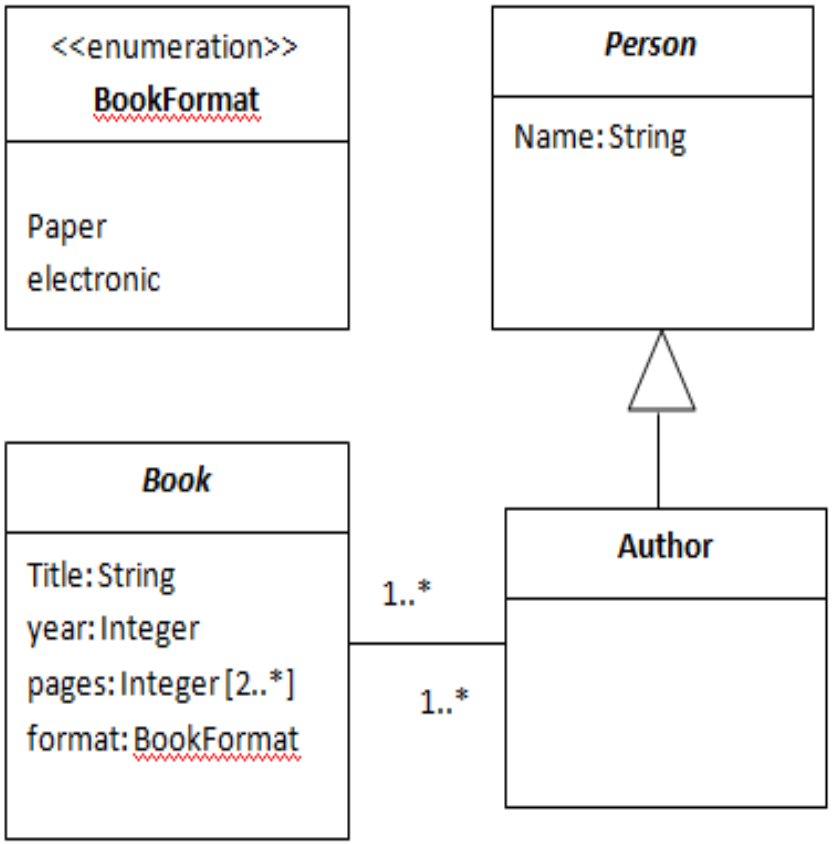
- Clafer (Class, feature, reference) is a general purpose lightweight modeling language.
- UML (Unified Modeling Language).

Motivation Example

1. A book has a title, and a year of publication.
2. It could have two or more pages.
3. The book's format could be paper or electronic.
4. A book has one or more authors.
5. An author is a person who has a name.

Motivation Example

UML



CLAFER

```
abstract Book
    title: string
    year: integer
    pages 2..*
    xor Format
        paper
        electronic
    author -> Author +

abstract Author: Person
abstract Person
    name: string
```

Textual UML Model

Model Book
Enum Format ={paper, electronic}

Class Book
Attributes
Title : String
Pages: Integer [2..*]
bookFormat: Format
Year: Integer
End

Abstract class Person
Attributes
Name: String
End

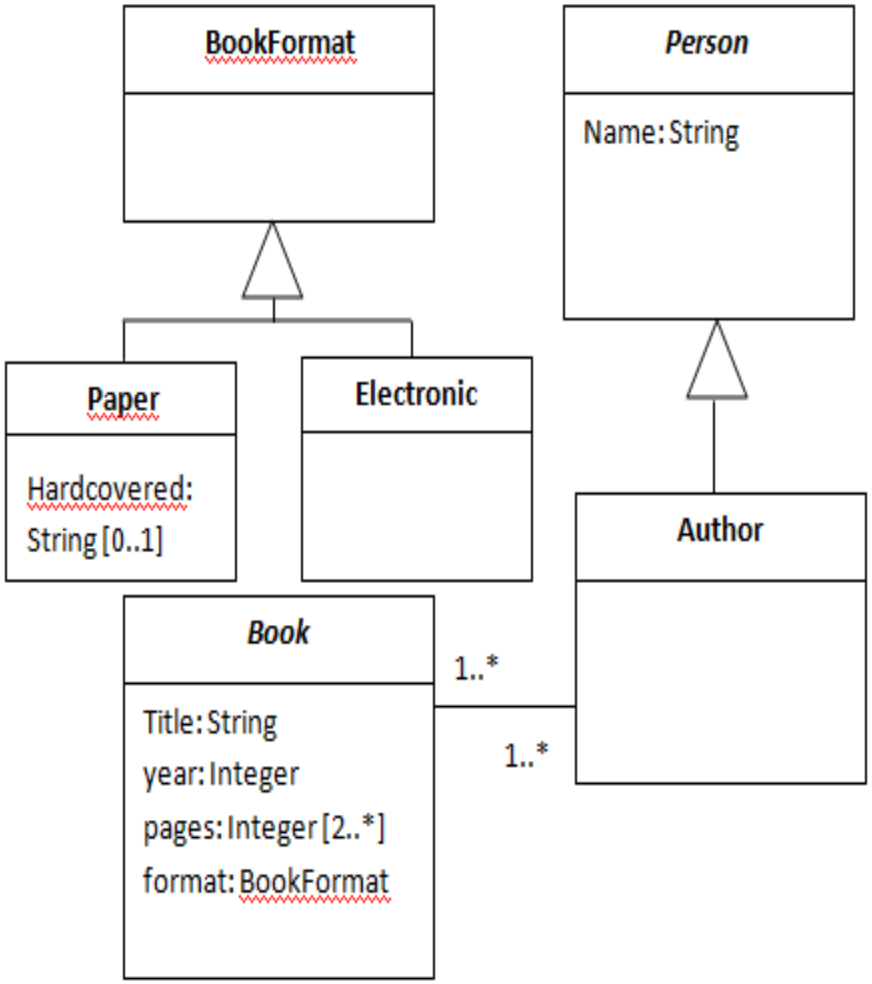
Class Author: Person

association BookAuthoredBy between:
Book [1] role authoredBy
Author[1..*] role authorOf
End

-- Constraints

Motivation Example

UML



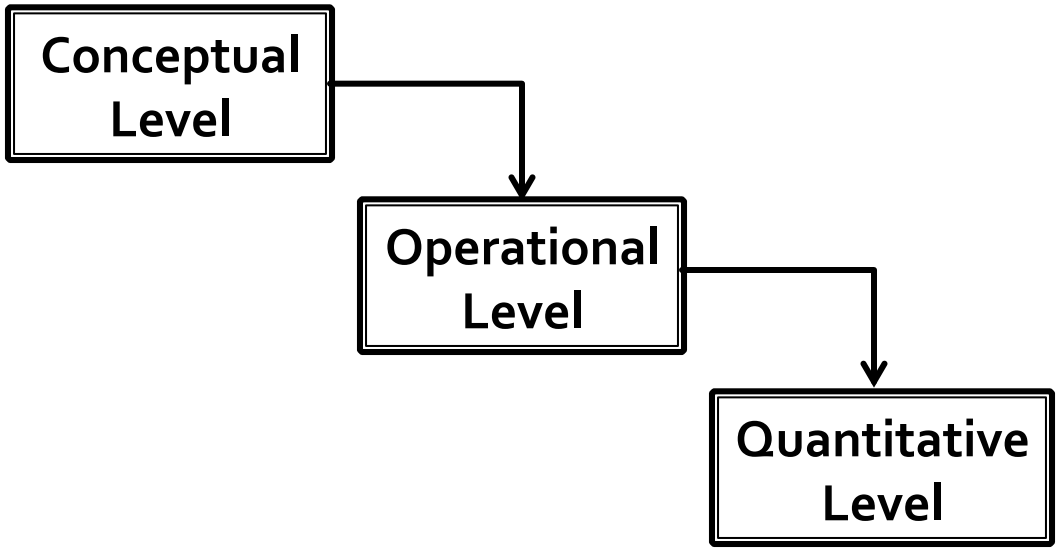
CLAFER

```
abstract Book
    title: string
    year: integer
    pages 2..*
    xor Format
        paper
                    hardcovered?
        electronic
    author -> Author +

abstract Author: Person
abstract Person
    name: string
```

GQM Methodology

Goal Question Metric Methodology (GQM)



Methodology

- Goal

"Design an empirical study to compare between Clafer and Textual UML in terms of their support for Structure Model Evolution at the Requirements Stage"

Methodology

- Research Questions
  - Which language better supports structure modeling and expressing constraints?
  - Which language better supports Model Evolution? What are the strengths and weaknesses of each language?
  - What kind of tool support would help with Model Evolution?
  - What are the recommendations for the design of Clafer?

Methodology

- Choice of Textual UML model specification.
  - Kernel MetaMetaModel (KM3) specification.
  - USE specification.
- Extensions to the USE specification.
- Pilot Study as a first step towards an empirical one.

Methodology

- Choice of the system to be modeled:
  - ORACLE Retail Merchandize Financial (Oracle Retail MFP) system.
- Formalizing a numbered Requirements Document.

Evaluation Criteria

QUESTIONS

1. Which language better supports structure modeling and expressing constraints?

2. Which language better supports Model Evolution? What are the strengths and weaknesses of each language?

METRICS

- Expressiveness (1)
- Model Size (1)
- Requirements Distribution (1, 2)
- Amount of Restructuring (2)
- Locality of Change (2)
- Frequency of Errors (2)
- Redundancy (1, 2)
- Validation Mechanisms (2)
- Speed of Modeling (1, 2)

Evaluation Criteria

- **Model Size**
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Frequency of Errors
- Redundancy
- Validation Mechanisms
- Speed of Modeling

Analysis and Evaluation

- **Model Size:**

	Clafer	UML
Total Number of Lines of Code (without namespaces or comments)	150	250
Total Number of characters (without spaces)	4647	8456

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Frequency of Errors
- Redundancy
- Validation Mechanisms
- Speed of Modeling

Evaluation Criteria

- Model Size
- **Expressiveness**
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Frequency of Errors
- Redundancy
- Validation Mechanisms
- Speed of Modeling

Analysis and Evaluation

- **Clafer:**

```
abstract metric
        xor metricName
                xor sales
                        regularSales
                        promotionalSales
                        clearanceSales
                xor markdown
                        regularMarkdown
                        promotionalMarkdown
                        permenantMarkdown
                turn
                receipts
                inventory
                grossMargin
                openToBuy
```

Analysis and Evaluation

- **UML:**
  - Alternative 1:

```
enum Metric { regularSales, promotionalSales, clearanceSales,
        regularMarkdown, promotionalMarkdown, permenantMarkdown,
        turn, receipts, inventory, grossMargin, openToBuy}
```

  - Alternative 2:

```
context metric
inv CheckSales:
        self.regularSales.isDefined() implies
        self.metricName =#Sales
```

Analysis and Evaluation

CLAFER

- Nesting denotes a relation.

```
xor measureType
        referenceMeasure
        nonReferenceMeasure
                editedBy -> planningRole ?
```

TEXTUAL UML

- Explicit OCL Constraints.

```
enum MeasureFormat {Reference, NonReference}
```

```
class Measure
attributes
        measureType: MeasureFormat
end
```

```
association RoleEditsMeasures between
Measure [1..*]          role EditedMeasure
PlanningRole[1]         role EditedBy
end
```

```
context Measure
inv EditedMeasureCondition:
        self.EditedBy.isDefined()   implies
        self.measureType =#NonReference
```

Analysis and Evaluation

CLAFER

- Contextual Name Lookup

```
xor viewType
      Approval
            [ specificRole.roleLevel = MiddleOut ]
```

TEXTUAL UML

- Checking the value of each variable sequentially.

```
inv ApprovalView:
self.Type =#Approval  implies
self.visibility =#SpecificRole and
self.planningRole -> exists (roleLevel =#MiddleOut)
```

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Frequency of Errors
- Redundancy
- Validation Mechanisms
- Speed of Modeling

Evaluation Criteria

- Model Size
- Expressiveness
- **Requirements Distribution**
- Amount of Restructuring
- Locality of Change
- Frequency of Errors
- Redundancy
- Validation Mechanisms
- Speed of Modeling
- Desugaring

Analysis and Evaluation

- Clafer Modeling

```
abstract view
	viewBelongsTo -> workflow
	xor visibility
		specificRole -> planningRole
			[ viewBelongsTo.workflowPlanningSeason.inSeasonPlanning => ! specificRole.roleLevel = TopDown ]
		allRoles
			roles -> planningRole * = planningRole

	measures -> Measure +

	planversion -> plan
	[ viewBelongsTo.workflowPlanningSeason.inSeasonPlanning => ! (planversion = WaitingForApproval || planversion = LastYear)]

	xor viewType
		Approval
			[ specificRole.roleLevel = MiddleOut ]
			[ planversion = WaitingForApproval ]
			[ viewBelongsTo.workflowPlanningSeason.preSeasonPlanning]
		Seeding
			[ viewBelongsTo.workflowPlanningSeason.preSeasonPlanning]
			[ specificRole.roleLevel =TopDown || specificRole.roleLevel = MiddleOut]
			[ planversion = WorkingPlan ]

		Review
		Planning
```

Analysis and Evaluation

- Textual UML Modeling

```
-----------------------------------------------------------------------

enum ViewVisibility {SpecificRole, AllRoles}
enum ViewType {Approval, Seeding, Review, Planning}

-----------------------------------------------------------------------

class View
attributes
        visibility: ViewVisibility
        Type: ViewType

end

-----------------------------------------------------------------------

association RolesAndViews between
View [1..*]
PlanningRole [1..*]
end

association PlansAssociatedWithViews between
Plan[1..*]
View[1..*]
end

aggregation ViewHasMeasures between
view [1..*]
Measure [1..*]
end

composition WorkFlowHasViews between
Workflow [1]
view[1..*]
end
```

Scroll Up 120 Lines

Start

Scroll Down 70 lines from the Start point

Analysis and Evaluation

- Textual UML Modeling

```
context View
inv SpecificRoleVisibility:
	self.workflow.workflowPlanningSeason =#InSeasonPlanning and
	self.visibility =#SpecificRole implies
	(not (self.planningRole ->exists (roleLevel=#TopDown)))

inv AllRolesVisibility:
	self.visibility =#AllRoles implies
	self.planningRole -> includesAll(self.planningRole)

inv viewSeasonPlans:
	self.workflow.workflowPlanningSeason =#InSeasonPlanning implies
	(not (self.plan -> exists(oclIsTypeOf(WaitingForApprovalPlan) or oclIsTypeOf(LastYearPlan))))

inv ApprovalView:
	self.Type =#Approval  implies
	self.visibility =#SpecificRole and
	self.planningRole -> exists (roleLevel =#MiddleOut) and
	self.workflow.workflowPlanningSeason =#PreSeasonPlanning and
	self.plan -> exists(oclIsTypeOf(WaitingForApprovalPlan))

inv SeedingView:
	self.Type =#Seeding  implies
	self.visibility =#SpecificRole and
	(self.planningRole -> exists (roleLevel =#TopDown) or self.planningRole -> exists (roleLevel =#MiddleOut)) and
	self.workflow.workflowPlanningSeason =#PreSeasonPlanning and
	self.plan -> exists(oclIsTypeOf(WorkingPlan))
```

Scroll Down further 120 lines

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Redundancy
- Frequency of Errors
- Validation Mechanisms
- Speed of Modeling

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- **Amount of Restructuring**
- **Locality of Change**
- Redundancy
- Frequency of Errors
- Validation Mechanisms
- Speed of Modeling

Analysis and Evaluation

- Requirement #4: There are three types of planning roles in MFP; Top-down, Middle-out, and Bottom-up.

Clafer

```
abstract planningRole
        xor roleLevel
                TopDown
                MiddleOut
                BottomUp
```

UML

```
enum PlanningRoleLevel { TopDown, MiddleOut, BottomUp}

----------------------------------------------------------

class PlanningRole
attributes
roleLevel: PlanningRoleLevel


end
```

Analysis and Evaluation

- Requirement #5: Top-down roles create the overall targets for the company and set top-down targets for the middle out role.

Clafer

```
abstract planningRole
        xor roleLevel
                TopDown
                        createTopDownTargets -> Target +
                MiddleOut
                        receivesTopDownTargets -> Target +
                BottomUp

abstract Target
        createdBy -> planningRole
        [createdBy.roleLevel.TopDown => publishedTo.roleLevel.MiddleOut]

        publishedTo -> planningRole
```

Analysis and Evaluation

UML

```
enum PlanningRoleLevel { TopDown, MiddleOut, BottomUp}
```

```
class PlanningRole
attributes
roleLevel: PlanningRoleLevel

end

class Target
attributes

end
```

```
association TargetCreatedBy between
Target [1..*]       role  createdTarget
PlanningRole [1]    role  TargetCreator
end

association TargetPublishedTo between
Target [1..*]       role  publishedTarget
PlanningRole [1]    role  TargetReceiver
end
```

```
context Target
inv TargetsPublishedTo:
        self.TargetCreator.roleLevel = #TopDown implies
        self.TargetReceiver.roleLevel = #MiddleOut
```

A total of **10** lines were added

Scrolling through **100** lines

A more complicated OCL constraint.

Analysis and Evaluation

- Requirement #6: Middle-out roles create middle-out targets.

Clafer

```
abstract planningRole
        xor roleLevel
                TopDown
                        createTopDownTargets -> Target +
                MiddleOut
                        receivesTopDownTargets -> Target +
                        createMiddleOutTargets -> Target +
                BottomUp
```

Analysis and Evaluation

- Requirement #8: The targets are published by superior levels to the subsequent level: top down passes to middle out, and middle out passes to bottom up.

```
abstract planningRole
        xor roleLevel
                TopDown
                        createTopDownTargets -> Target +
                MiddleOut
                        receivesTopDownTargets -> Target +
                        createMiddleOutTargets -> Target +
                BottomUp
                        receivesMiddleOutTargets -> Target +
                        createPlans -> plan +

abstract Target
        createdBy -> planningRole
        [ createdBy.roleLevel.TopDown =>
                publishedTo.roleLevel.MiddleOut ]
        [ createdBy.roleLevel.MiddleOut =>
                publishedTo.roleLevel.BottomUp ]

        publishedTo -> planningRole
```

Analysis and Evaluation

UML added parts

```
association RoleCreatesPlans between
PlanningRole [1..*]
Plan [1..*]
end
-----------------------------------------------------------------------------------------------------
context Target
   inv TargetsCreatedBy:
        self.TargetCreator.roleLevel = #TopDown or
        self.TargetCreator.roleLevel = #MiddleOut

inv TargetsPublishedTo:
        self.TargetCreator.roleLevel = #TopDown implies
        self.TargetReceiver.roleLevel = #MiddleOut or
        self.TargetCreator.roleLevel = #MiddleOut implies
        self.TargetReceiver.roleLevel = #BottomUp


inv TargetsNotPublishedTo:
        self.TargetReceiver.roleLevel = #MiddleOut or
        self.TargetReceiver.roleLevel = #BottomUp
```

More amount of restructuring in UML.

Restructuring is not done in the same context/part of the model.

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Redundancy
- Frequency of Errors
- Validation Mechanisms
- Speed of Modeling

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- **Redundancy**
- Frequency of Errors
- Validation Mechanisms
- Speed of Modeling

Analysis and Evaluation

- Clafer:

```
abstract planningRole
        roleCreatedTheFollowingTargets -> Target *
        [roleLevel=BottomUp => no roleCreatedTheFollowingTargets]

abstract Target
        createdBy -> planningRole
        [! createdBy.roleLevel = BottomUp]
```

Analysis and Evaluation

- Redundancy is mostly present in constraints.

Clafer Modeling

```
abstract WorkingPlan: plan
        seeded?
                seedingSource -> LastYear
                lastSeedingDate: string
                seededWithThoseInstances -> HierarchyLevelInstance *
```

UML Modeling

```
context WorkingPlan
inv SeedingSourceExists:
        self.seedingSource.isDefined() implies self.seeded.isDefined()

inv SeedingDateExists:
        self.lastSeedingDate.isDefined() implies self.seeded.isDefined()

inv SeedingInstancesExist:
        self.possibleSeedingSources -> exists(isDefined) implies self.seeded.isDefined()
```

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Redundancy
- Frequency of Errors
- Validation Mechanisms
- Speed of Modeling

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Redundancy
- **Frequency of Errors**
- Validation Mechanisms
- Speed of Modeling

Analysis and Evaluation

- **Clafer:**

Type of Error	Frequency
Missing References	Highest (about 6 times)
Missing Constraints	About 4 times
Syntax errors accessing enumerations	twice

Analysis and Evaluation

- **UML:**

Type of Error	Frequency
OCL Constraints	14 times
Mistakes in handling associations.	1 time; Workflow Requirements
Syntax errors accessing enumerations	6 times

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Redundancy
- Frequency of Errors
- Validation Mechanisms
- Speed of Modeling

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Redundancy
- Frequency of Errors
- **Validation Mechanisms**
- Speed of Modeling

Analysis and Evaluation

```
abstract Target
        belongsTo -> Targetplan

        createdBy -> planningRole
        [! createdBy.roleLevel = BottomUp]

        publishedTo -> planningRole
        [ createdBy.roleLevel=TopDown => publishedTo.roleLevel = MiddleOut ]
        [ createdBy.roleLevel=MiddleOut => publishedTo.roleLevel = BottomUp ]


//********** Concrete Clafers ************************************************

TopDownSalesTargets2012: Target
        [  belongsTo = TargetPlan2012
           createdBy = ExecutiveManager
           publishedTo = PlanningDirector ]
```

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Redundancy
- Frequency of Errors
- Validation Mechanisms
- Speed of Modeling

Evaluation Criteria

- Model Size
- Expressiveness
- Requirements Distribution
- Amount of Restructuring
- Locality of Change
- Redundancy
- Frequency of Errors
- Validation Mechanisms
- **Speed of Modeling**

Analysis and Evaluation

	Clafer	UML
Requirements 23, 24	2 hours	1 hour , 15 minutes
Requirement 11	2 hours	2 hours, 30 minutes

Clafer Modeling of Requirement #11

```
abstract HierarchyLevel
        levelBelongsToThis -> Hierarchy
        spreadsToLowerLevel -> HierarchyLevel ?
        aggregatesToHigherLevel -> HierarchyLevel ?
        [ all disj d1; d2 : HierarchyLevel |
        (d2 = d1.aggregatesToHigherLevel <=> d1 = d2.spreadsToLowerLevel) ]
```

Analysis and Evaluation

UML Modeling of Requirement #11

```
class HierarchyLevel
attributes

end
-----------------------------------------------------------------------------------------------
association LevelSpreadsTo between
HierarchyLevel[1]        role   base
HierarchyLevel[0..1]     role   lowerLevel
end

association LevelAggregatesTo between
HierarchyLevel[1]        role   currentLevel
HierarchyLevel[0..1]     role   higherLevel
end
-----------------------------------------------------------------------------------------------
context HierarchyLevel
inv AggregatesTo:
        HierarchyLevel.allInstances()->
        forAll(l1, l2| l1.higherLevel=l2 implies l2.currentLevel=l1)

inv SpreadsTo:
        HierarchyLevel.allInstances()->
        forAll(l1, l2| l1.lowerLevel=l2 implies l2.base=l1)
```

Advantages

CLAFER

1. Clafer's support for hierarchy/Nesting improves the expressiveness of the model.
2. Name Lookup helps keeping the constraints simpler and more concise.
3. Concrete Clafers provide a good validation mechanism.

USE SPECIFICATION

1. Associations in UML help in preventing redundancy.
2. Simpler and more clear syntax for Class declarations/ Associations, and enumeration definitions.
3. Better visualization of association multiplicities.

Advantages

CLAFER

4. Succinct model representation
5. Better support for locality of change
6. Less restructuring amount for evolution steps.

USE SPECIFICATION

4. More supported Data Types.

Disadvantages

CLAFER

1. Taking care of inverses introduces errors and redundancy in the models.
2. Absence of associations make it less readable to visualize associations ends and their multiplicities.

UML

1. Complicated OCL constraints representations due to large number of operators, large number of sequence step.
2. Distribution of modeling sections decreases the support for locality of change.
3. Absence of an integrated validation mechanism.

Threats to Validity

- Modeling was only carried out by one person.
- Previous knowledge of UML Class Diagrams.
- Object Learning Effect.
- Different training and reading materials.
- Having Clafer developers around.

Summary & Future Work

Summary

- Formalized the different types of constructs used by each language for Structure Model Evolution.
- Formalized a strong set of evaluation criteria for comparison.
- Preliminary results about the strengths and weaknesses of each language with regard to Model Evolution Support.

Future Work

- Features to be explored:
  - Local and Global Constraints in Clafer versus UML.
  - Difficulty of representing Group Cardinality in UML as OCL constraints.
  - Name Lookup feature in Clafer and its effect on writing constraints in deep hierarchies.

Future Work

- Experimental Design Considerations:
  - Subjects should only model the problem using one modeling language to avoid the Object Learning Effect.
  - Keep randomization of requirements to have unbiased results.
  - Ask subjects to commit a version after modeling each requirement.
  - Proper training materials.

Future Work

- Background Questionnaire.
- No tool support.
- Additional Evaluation Criteria:
  - Number of different language constructs used.
  - Familiarity of Notation or the Speed of becoming proficient with a given language.
  - Confidence/Comfort level.
  - Keeping track of Mistakes to give insight what kind of tool support is needed.

Thank You