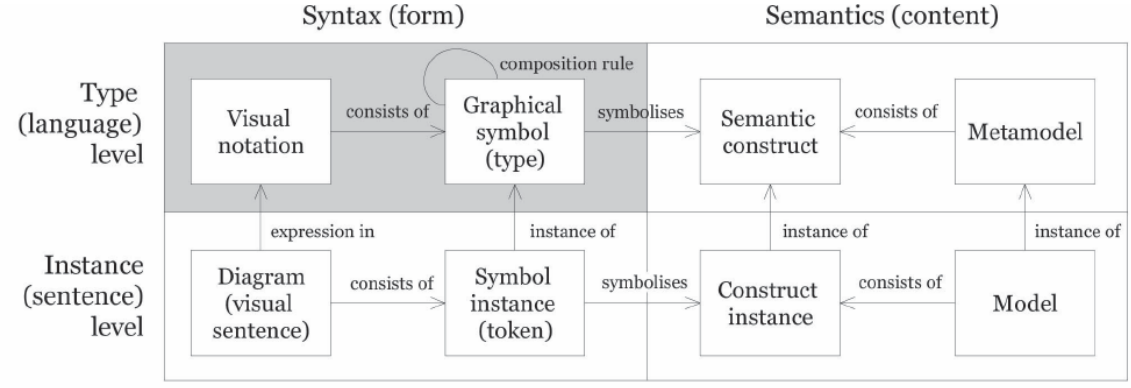
COMPSCI 732 Software Tools
Visual Languages: Design and Evaluation

- Topics:
  - Elements of a visual language solution
    - Surface notation
    - Meta model and meta modelling
    - Semantic processing (see later in Karen's section)
  - Design and evaluation of visual languages
    - Techniques for designing "good" VLs
    - Cognitive dimensions
    - Physics of notations

Elements of a visual language solution

Surface VL defn

Meta model/information model defn



A VL diagram

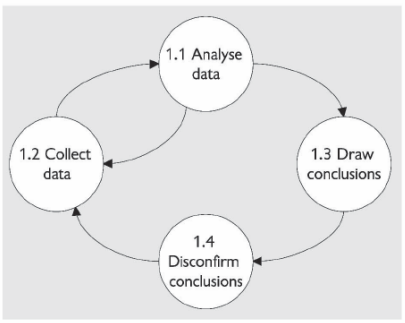
"Execution" of the diagram

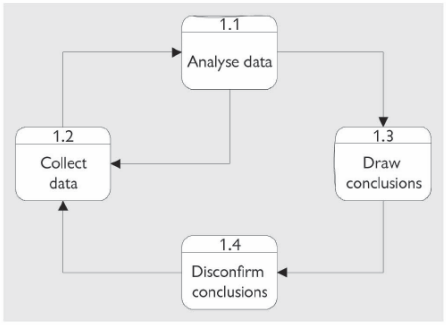
From Moody "Physics of Notations"

Surface notation

- The surface form of the language
- May be variants/dialects

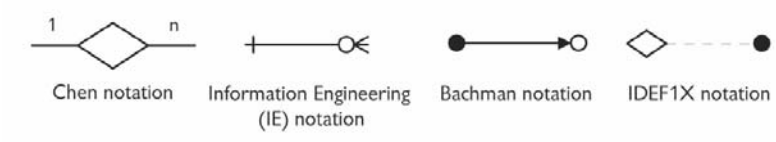
DeMarco DFD





Gane & Sarson DFD

ER Variants

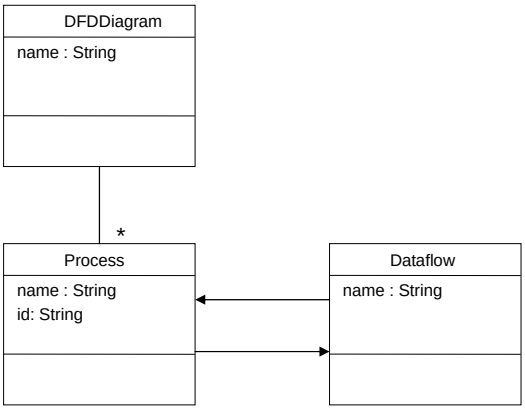


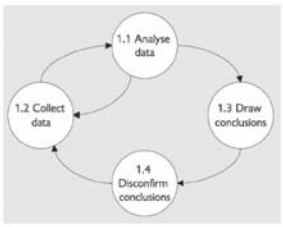
From Moody

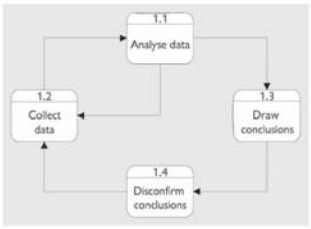
Meta model

- Describes underlying information model of the language
  - It's abstract syntax
  - "A metamodel is a precise definition of the constructs and rules needed for creating semantic models" Ernst
- Want to be able to describe precisely in the same way as we can describe formal models for textual languages
- Need mappings from visual syntax to meta model
  - From these define the equivalent of parsers

Simple DFD Meta Model







Meta modelling

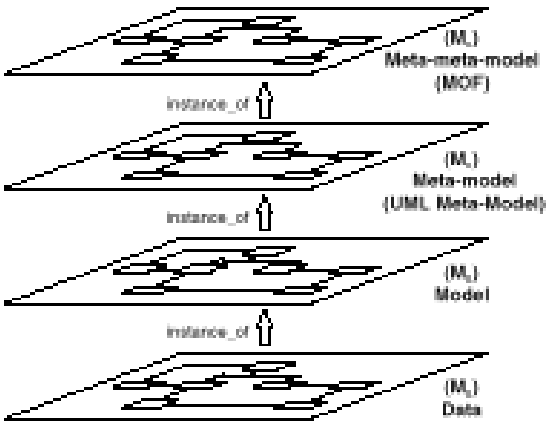
- Methodologies and techniques for deriving a meta model
  - Meta modelling deals with modelling of models
  - Bottom up – derive modelling needs in the form of a meta model (eg to configure a framework or product line) then a VL to express that
  - Top down – have a VL (eg one familiar to end users) and derive a meta model from this to build a tool to support modelling in that VL
- So what are metamodels good for:
  - As a "schema" for semantic data that needs to be exchanged.
  - As a schema for semantic data that needs to be stored, in a repository, for example.
  - **As a language that supports a particular methodology or process** - a metamodel allows a language designer or methodologist to better capture, analyze and understand what they are actually writing about in all those methodology books.
  - As a language to express additional semantics of existing information.

Meta modelling

- Need a notation for expressing the meta model!
  - Has both structural aspects (eg generalisation) and behavioural (constraints of various sorts)
  - Typically use a constrained form of UML class diagrams plus OCL
    - eg the Meta Object Facility MOF
    - eCore is a commonly used variant of MOF
    - eCore support provided in Eclipse
  - But how do you define that?
    - Meta meta model using MOF itself

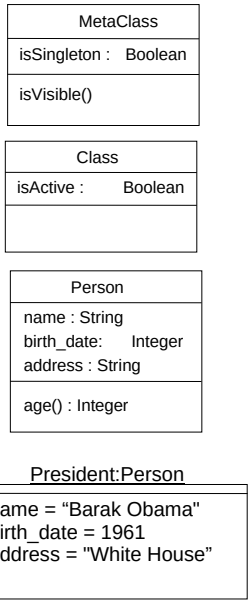
4 layer model – eg for UML

- This approach leads to a four layer approach to the modelling
- meta-meta-model (M3): defines the MOF notation
- meta-model (M2): defines UML notation using MOF
- user model (M1): a UML model of a particular problem domain
- data (M0): typical objects instantiating the UML model
- Note: could use M3 instead to define M2 for ER modelling; M1 a typical ER model; M0, typical ER data.



Examples

- Typical examples of elements at each level:
- M3: MOF MetaClass
- M2: UML Class, instance of MOF Class; very similar to MOF concept of a Class
- M1: Person, a typical instance of UML Class
- M0: President:Person, a typical instance of Class Person
- From C. Atkinson, Supporting and applying the UML conceptual framework.



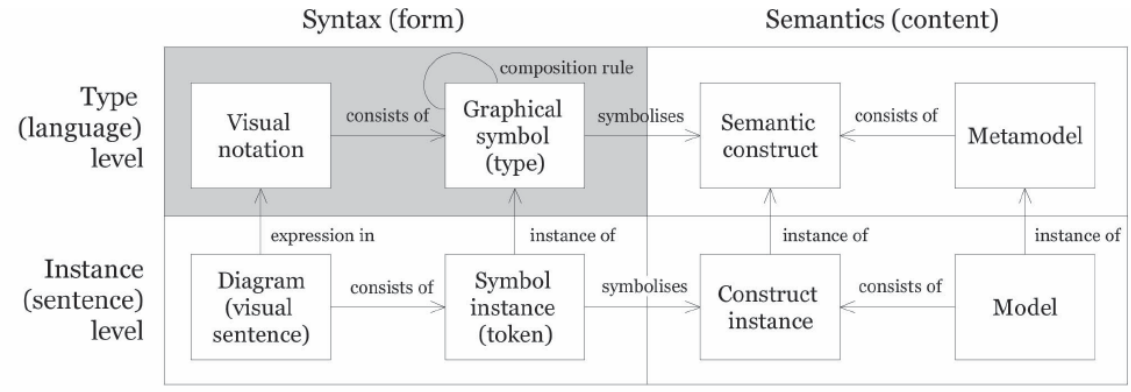
Most people don’t model at this level

If defining a UML tool model this

People using the tool model this

At “runtime” may result in these

VL design and evaluation



Cognitive Dimensions Framework

- Green and Petre 1996 (since developed by Blackwell)
- Establishes a set of "dimensions" to think about the tradeoffs made in implementing visual programming environments
  - Has had very strong influence on the VL community
  - Means of explaining effects of design decisions
- Comes out of cognitive psychology community
- Lightweight – doesn't need large usability studies to get useful insight
- Can be used for evaluation and also as a design aid

Cognitive Dimensions

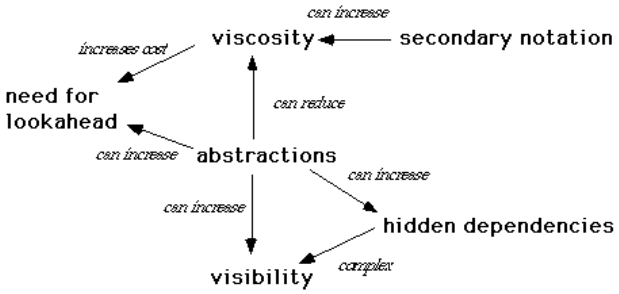
- **Abstraction gradient What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?**
- **Closeness of mapping What 'programming games' need to be learned?**
- **Consistency When some of the language has been learnt, how much of the rest can be inferred?**
- **Diffuseness How many symbols or graphic entities are required to express a meaning?**
- **Error-proneness Does the design of the notation induce 'careless mistakes'?**
- **Hard mental operations Are there places where the user needs to resort to fingers or penciled annotation to keep track of what's happening?**
- **Hidden dependencies Is every dependency overtly indicated in both directions? Is the indication perceptual or only symbolic?**

Cognitive Dimensions

- **Premature commitment Do programmers have to make decisions before they have the information they need?**
- **Progressive evaluation Can a partially-complete program be executed to obtain feedback on "How am I doing"?**
- **Role-expressiveness Can the reader see how each component of a program relates to the whole?**
- **Secondary notation Can programmers use layout, color, or other cues to convey extra meaning, above and beyond the 'official' semantics of the language?**
- **Viscosity How much effort is required to perform a single change?**
- **Visibility Is every part of the code simultaneously visible (assuming a large enough display), or is it at least possible to compare any two parts side-by-side at will? If the code is dispersed, is it at least possible to know in what order to read it?**

Use of Cognitive Dimensions

- Note the tradeoffs that occur
  - May add an abstraction that makes it easier to change things (reduced viscosity) but increases the difficulty of understanding (increased abstraction gradient and increased hidden dependencies).
  - See Green and Petre paper for several examples illustrating tradeoffs made



- Burnett provides a set of representation benchmarks that assist in operationalising the use of the CD framework.

Exercise

- CD evaluation of UML class diagrams
- Look at tradeoffs made
- Look at how designing appropriate tool support could mitigate problems

Cognitive Dimensions provides vocabulary

Verbatim transcript from a newsgroup discussion (real words from real users).

NB: this discussion referred to a version of Framemaker that is now obsolete.

- *A:* ALL files in the book should be identical in everything except body pages. Master pages, paragraph formats, reference pages, should be the same.
- *B:* Framemaker does provide this ... File -> Use Formats allows you to copy all or some formatting categories to all or some files in the book.
- *A:* Grrrrrrrrr …….. Oh People Of Little Imagination !!!!!!
- Sure I can do this ... manually, every time I change a reference page, master page, or paragraph format …..
- What I was talking about was some mechanism that automatically detected when I had made such a change. (…..) Or better yet, putting all of these pages in a central database for the entire book ……
- *C:* There is an argument against basing one paragraph style on another, a method several systems use. A change in a parent style may cause unexpected problems among the children. I have had some unpleasant surprises of this sort in Microsoft Word.

Improved Discussion

- *A:* Framemaker is too viscous.
- *B:* With respect to what task?
- *A:* With respect to updating components of a book. It needs to have a higher abstraction level, such as a style tree.
- *C:* Watch out for the hidden dependencies of a style tree.
- *(further possible comments)*
- The abstraction level will be difficult to master; getting the styles right may impose lookahead.

From: An Introduction to the Cognitive Dimensions Framework, T R G Green

http://homepage.ntlworld.com/greenery/workStuff/Papers/introCogDims/index.html

Attention Investment

- Theory to explain why people spend time doing programming
- Programming defined very broadly
- Defines “attention units”: nominal amount of “concentration” applied
- Applies a cost benefit analysis approach to programming activities
  - Programming => automation to save time in the future
  - Has Cost: attention units to do the job
  - Investment: attention units expended towards a potential reward
  - Pay-off: reduced future cost from investment
  - Risk: probability that no pay-off or –ve pay-off results
- See: First steps in programming: a rationale for attention investment models, Blackwell.

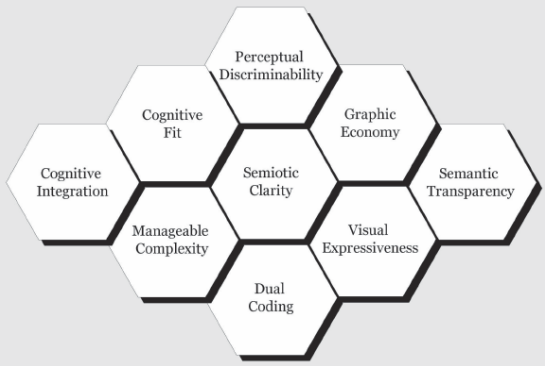
Champagne Prototyping

- A “cheap” method for early design evaluation
- Combines:
  - simple prototyping
    - used overlays and “look don’t touch” approach
  - cognitive walkthroughs with credible participants
  - cognitive dimensions & attention investment for analysis

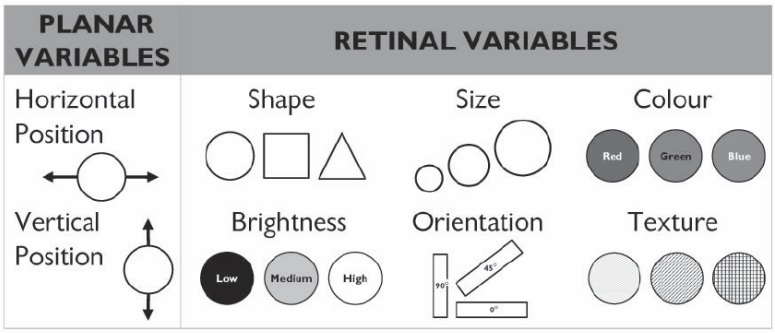
  to assist in answering questions at early design phase of visual environments
- Blackwell, Burnett and Peyton Jones, Champagne Prototyping: a research technique for early evaluation of complex end user programming systems, IEEE VL/HCC, 2004, 47-54

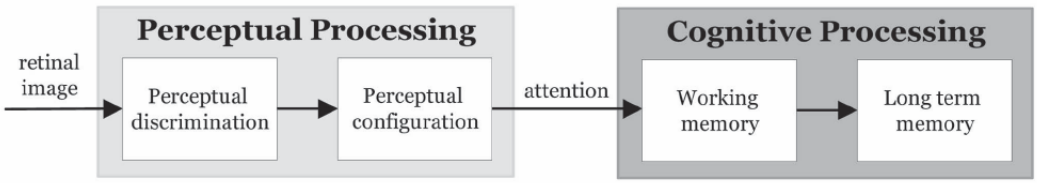
Physics of Notations: Moody

- Claims to be a "theory" that allows you to predict qualities of a visual notation
- A "meta study" – set of principles derived from synthesising multiple empirical results that others have reported
- New work, but will have significant influence in VL design



Design and solution space





Nine principles

- Principle of Semiotic Clarity: There should be a 1:1 correspondence between semantic constructs and graphical symbols
- Principle of Perceptual Discriminability: Different symbols should be clearly distinguishable from each other
- Principle of Semantic Transparency: Use visual representations whose appearance suggests their meaning
- Principle of Complexity Management: Include explicit mechanisms for dealing with complexity
- Principle of Cognitive Integration: Include explicit mechanisms to support integration of information from different diagrams
- Principle of Visual Expressiveness: Use the full range and capacities of visual variables
- Principle of Dual Coding: Use text to complement graphics
- Principle of Graphic Economy: The number of graphical symbols should be cognitively manageable
- Principle of Cognitive Fit: Use different visual dialects for different tasks and audiences

Examples

- Principle of Semiotic Clarity: There should be a 1:1 correspondence between semantic constructs and graphical symbols

  Symbol redundancy UML

  Symbol overload ArchiMate

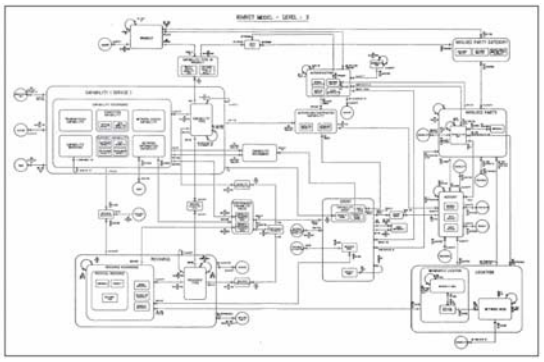
Examples

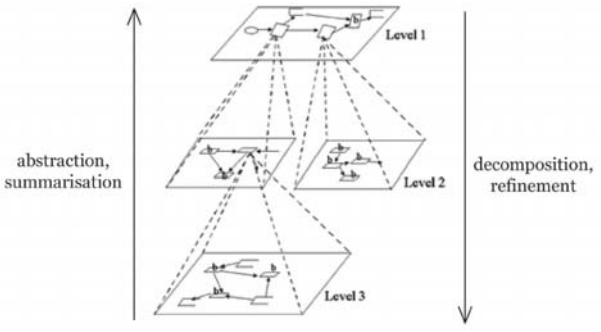
- Principle of Perceptual Discriminability: Different symbols should be clearly distinguishable from each other

  Visual distance important – shape, redundant coding important (cf visual expressiveness)

Examples

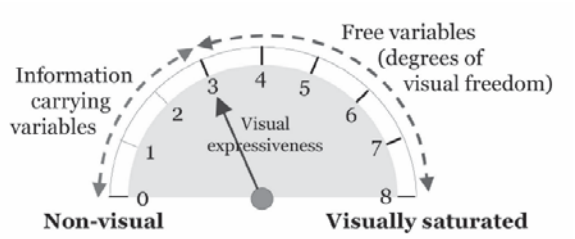
- Principle of Complexity Management: Include explicit mechanisms for dealing with complexity

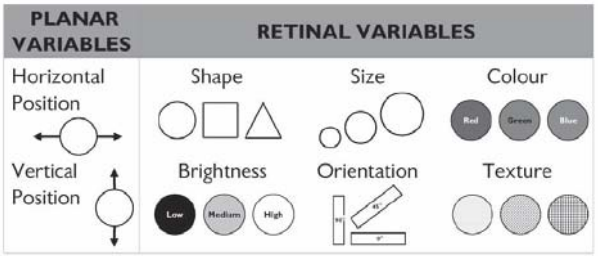


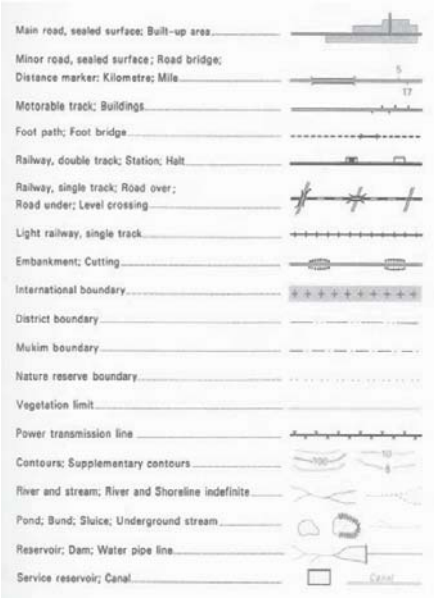


Examples

- Principle of Visual Expressiveness: Use the full range and capacities of visual variables







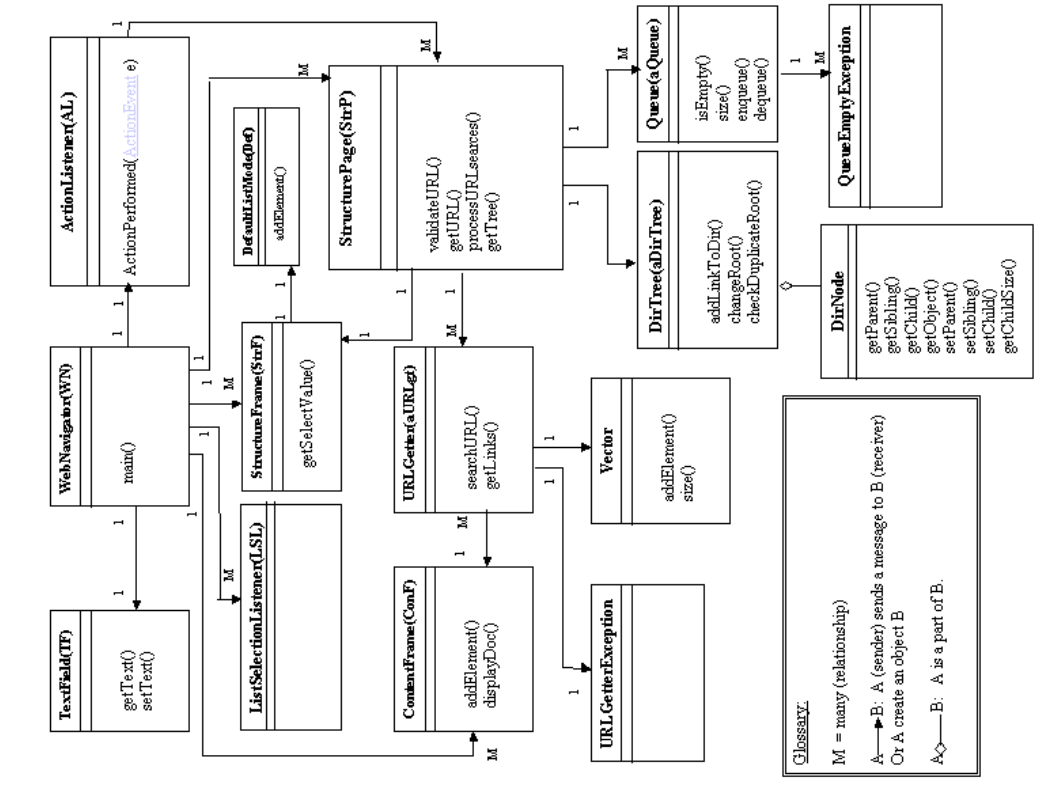
Exercise

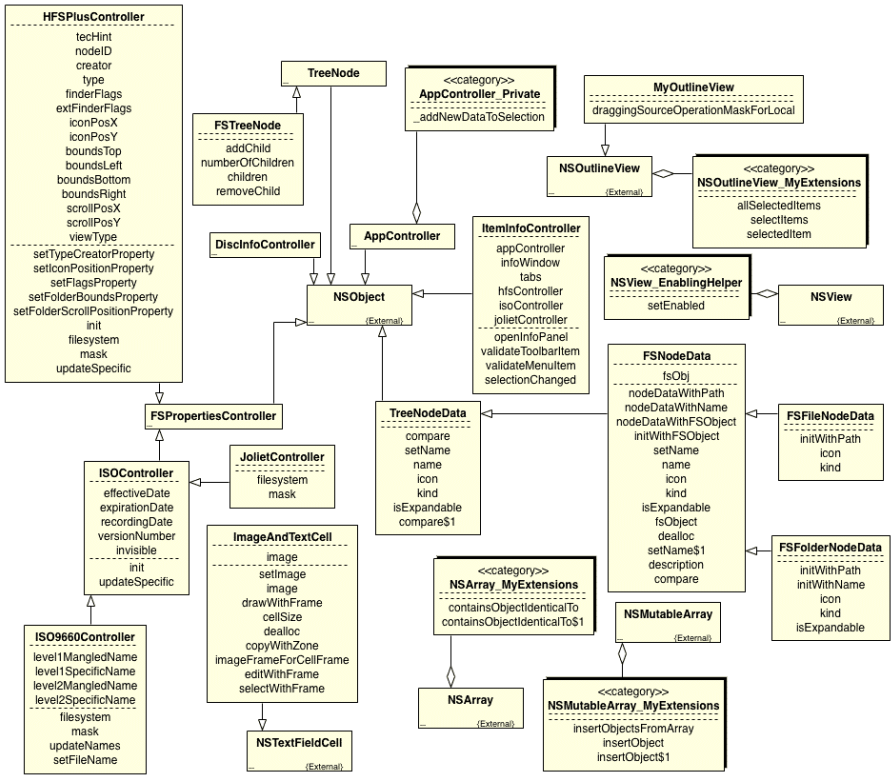
- Look at 2-3 of the principles applied to UML class diagrams
- Look at how changing design of UML surface notation could improve the language
- Look at how designing appropriate tool support could mitigate problems

Homework

- Thursday group activity will be in 279
- Undertake a CD and PON analysis of a visual language
- Will look at Swenson's Regatta
  - A Visual Language to Describe Collaborative Work, Swenson – see Resource page
  - Read this – make some preliminary notes about CD and PON wrt Regatta
  - Bring a copy of the paper plus CD and PON papers to class
  - Be prepared to argue your points

    - NB: if you are interested in Regatta, also look at our extensions of it: Low-level and high-level CSCW support in the Serendipity process modelling environment, Grundy et al.




HFSPlusController
tecHint
nodeID
creator
type
finderFlags
extFinderFlags
iconPosX
iconPosY
boundsTop
boundsLeft
boundsBottom
boundsRight
scrollPosX
scrollPosY
viewType
setTypeCreatorProperty
setIconPositionProperty
setFlagsProperty
setFolderBoundsProperty
setFolderScrollPositionProperty
init
filesystem
mask
updateSpecific
TreeNode
FSTreeNode
addChild
numberOfChildren
children
removeChild
<<category>>
AppController_Private
_addNewDataToSelection
MyOutlineView
draggingSourceOperationMaskForLocal
NSOutlineView
{External}
<<category>>
NSOutlineView_MyExtensions
allSelectedItems
selectItems
selectedItem
DiscInfoController
AppController
ItemInfoController
appController
infoWindow
tabs
hfsController
isoController
jolietController
openInfoPanel
validateToolbarItem
validateMenuItem
selectionChanged
NSObject
{External}
<<category>>
NSView_EnablingHelper
setEnabled
NSView
{External}
FSNodeData
fsObj
nodeDataWithPath
nodeDataWithName
nodeDataWithFSObject
initWithFSObject
setName
name
icon
kind
isExpandable
fsObject
dealloc
setName$1
description
compare
FSPropertiesController
TreeNodeData
compare
setName
name
icon
kind
isExpandable
compare$1
FSFileNodeData
initWithPath
icon
kind
JolietController
filesystem
mask
ISOController
effectiveDate
expirationDate
recordingDate
versionNumber
invisible
init
updateSpecific
ImageAndTextCell
image
setImage
image
drawWithFrame
cellSize
dealloc
copyWithZone
imageFrameForCellFrame
editWithFrame
selectWithFrame
FSFolderNodeData
initWithPath
initWithName
icon
kind
isExpandable
<<category>>
NSArray_MyExtensions
containsObjectIdenticalTo
containsObjectIdenticalTo$1
NSMutableArray
{External}
ISO9660Controller
level1MangledName
level1SpecificName
level2MangledName
level2SpecificName
filesystem
mask
updateNames
setFileName
NSArray
{External}
<<category>>
NSMutableArray_MyExtensions
insertObjectsFromArray
insertObject
insertObject$1
NSTextFieldCell
{External}