

iView MediaPro

a case study in conceptual modelling

Daniel Jackson
Lab in Software Engineering (6.170)
Lecture 22
October 30, 2002

why conceptual modelling?

conceptual models are 'sharp tools' [Brooks]

- › cheap to produce, but major impact on quality
- › benefit early, troublesome phases of spec & design

good CM brings

- › for specifier: grip on essence & extent of problem
- › for designer: cleaner abstractions, uniformity
- › for implementor: simpler interfaces, cleaner code
- › for user: easier grasp, predictability, ability to exploit fully

examples

concept nicely done

- › paragraph in Word
- › text frame in Quark Xpress
- › user in Unix

concept missing

- › face/font naming on all platforms
- › line-end features in Visio
- › discussion threads in email clients

concept hacked on

- › figures, sections, numbering in Word
- › form-filling in browsers
- › trash & sent folders in email clients

MediaPro: what it is

application for media files

- › photos, movies, fonts, audio, etc
- › organizing, viewing & exporting

history

- › started life as shareware in 1996
- › for multimedia CD-ROM
- › MediaPro written in Jan 2001
- › based in London, sales by net



MediaPro: why it's nice

small, simple & reliable

- › download is 2.4MB
- › easy to use, esp. getting started
- › rarely crashes or misbehaves

good performance

- › fast import, display, rotation, etc
- › handles large catalog (eg, 5000 photos)

good features

- › 120 file types
- › organization into 'sets'
- › batch operations (eg, rename files)

demo

basic operations

- › importing files
- › viewing, rotating, keywords

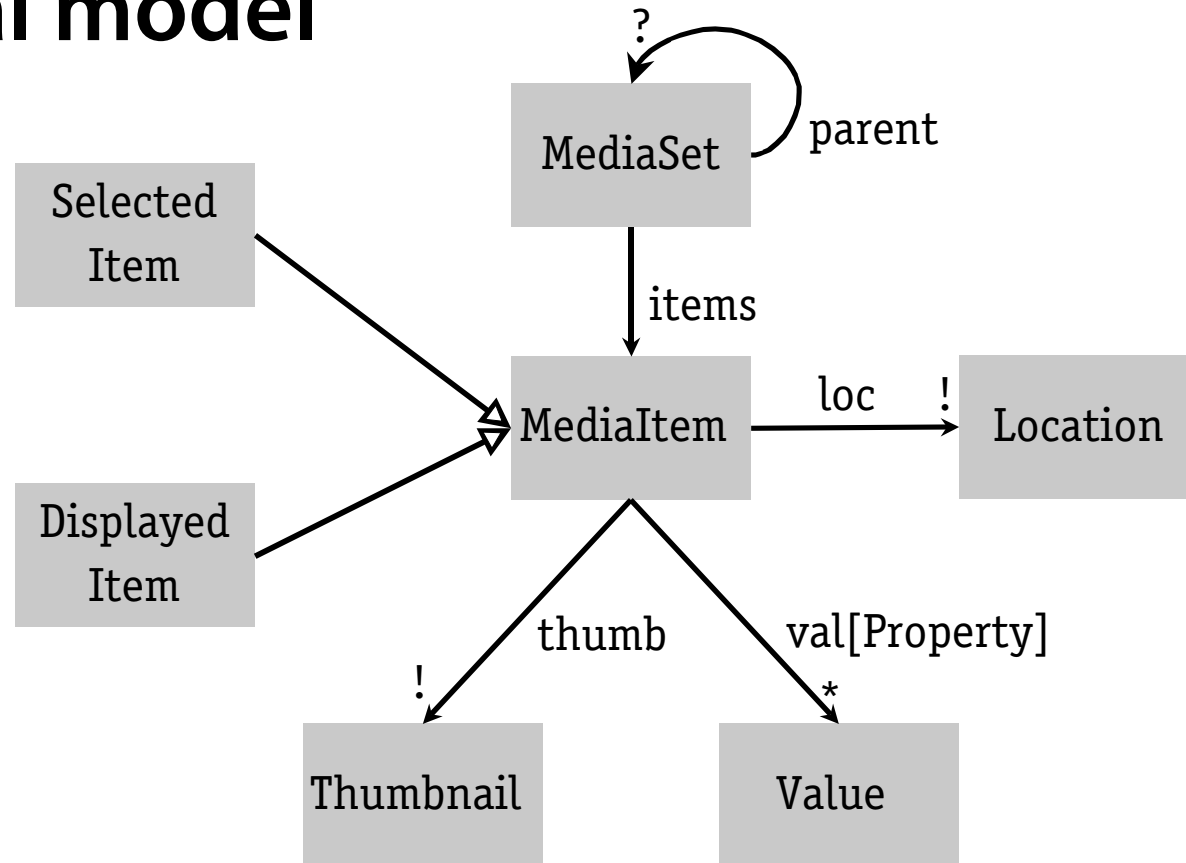
sets

- › selecting media files
- › creating sets explicitly
- › sets from folders
- › sets from keywords

updating against file system

- › exporting sets
- › finding missing items
- › importing again

conceptual model



MediaItem: an item stored by MediaPro

Location: path in filesystem

Property: of item, eg. label, keywords, date

loc: maps item to purported location

val: stored property/value mappings for item

an aside

suppose we

- › select some items
- › show selected
- › apply reorderings

how do these affect order in whole catalog?

a view problem!

- › just like subList in Java?

problems

relationship to file system based on files, not sets

- › watch folder, not watch tree
- › can't create folder tree to mirror set tree
- › reset path must be done file-by-file
- › find missing items ignores file moved to trash!

sets must be maintained manually

- › can add sets from fields (eg, keywords)
- › but can get out of sync
- › and what if set is changed manually?
(is order of items retained when added again?)

suggestion

- › need some new concepts

concept #1: defined sets

idea

- › classify some sets as 'defined'
- › user gives rules for how they're populated
- › make them unmodifiable views

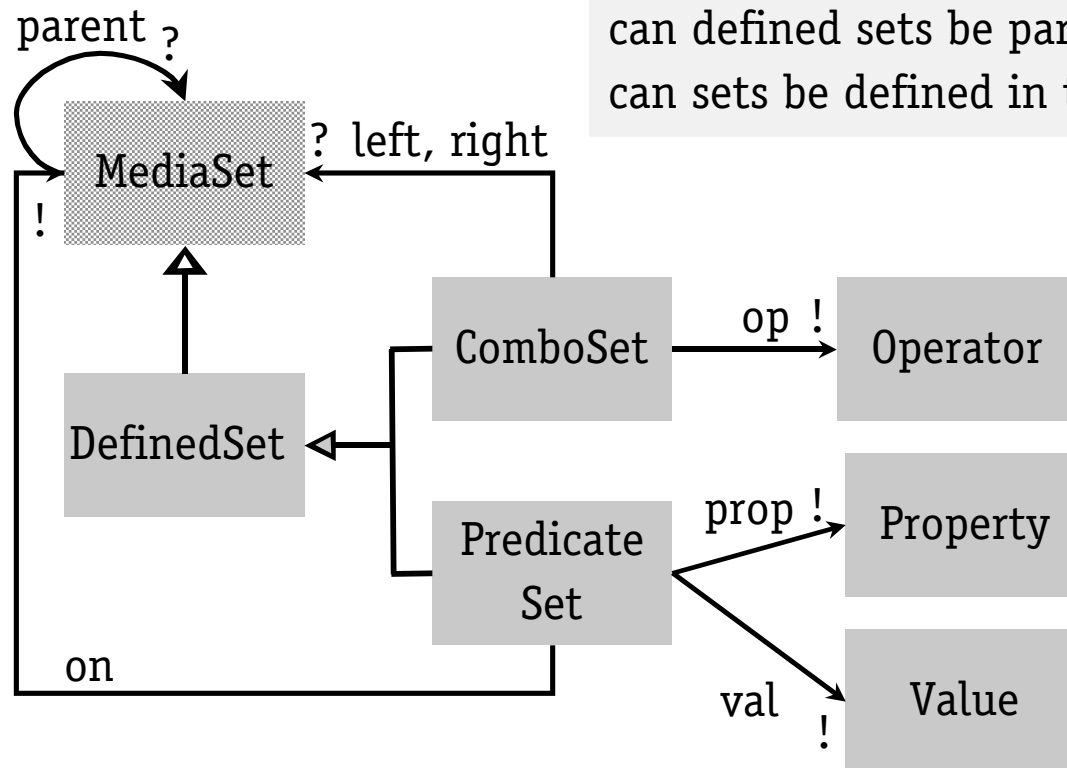
advantages

- › no complications from modify/add from field
- › kept in sync automatically
- › more powerful classification scheme

concept #1: defined sets

constraints

can defined sets be parents of other sets?
can sets be defined in terms of defined sets?



DefinedSet: a set based on another set by a definition rule; automatically updated

ComboSet: set defined to contain items as union, intersection, etc. of two other sets

PredicateSet: set defined to contain items that pass filter predicate, eg. keywords include “kids”

concept #2: synchronized sets

idea

- › mark some sets as synchronized with file system roots
- › offer one-way and two-way syncs as options?

advantages

- › can organize files offline
- › fewer problems with lost files
- › subsumes import, export, watch
- › can be used for backup
- › makes catalog more portable

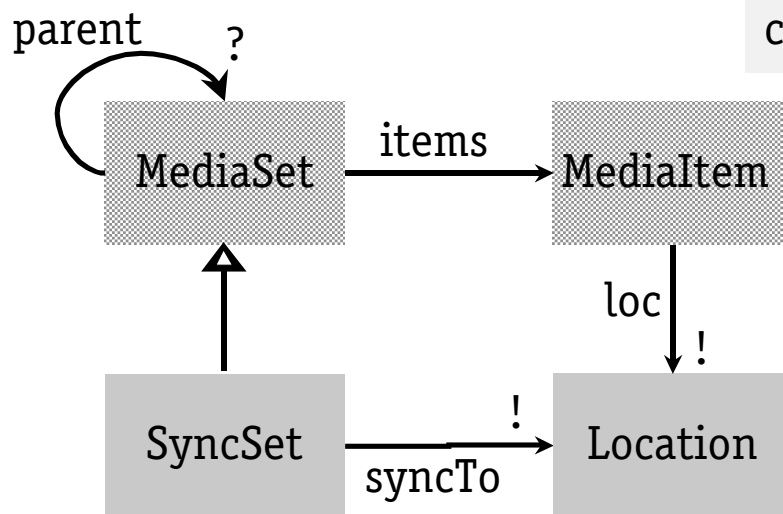
concept #2: synchronized sets

constraints

can sync sets have parents? children?

can an item be in more than one sync set?

can sync'd sets be defined sets?



SyncSet: a set that is kept in synchronization with a file system

syncTo: maps a SyncSet to the location of the root of the file system it's sync'd with

ramifications

defined sets

- › not hard to implement
- › reclassify items incrementally

synchronization

- › very tricky to get right
- › use off-the-shelf algorithm (eg, Unison)
- › judging dirtiness is tricky, and platform-dependent
- › generate action list for user approval

conclusions

conceptual modelling

- › is fun and useful
- › light on documentation
- › heavy on thinking

alloy project

- › a research project in LCS's software design group
- › alloy language: a textual notation, expresses dynamics too
- › alloy analyzer: fully automatic analysis using SAT

for your final project

- › construct CM's as needed
- › write them down!
- › waterfall approach not necessary: can do CM any time