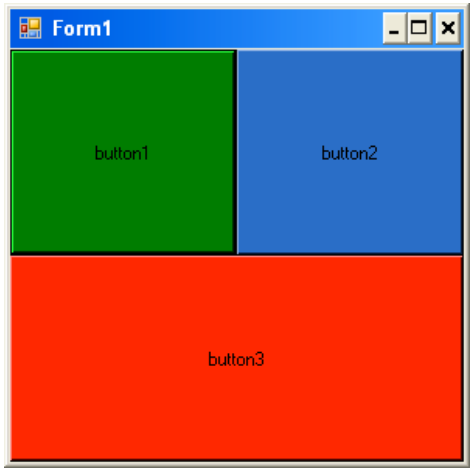
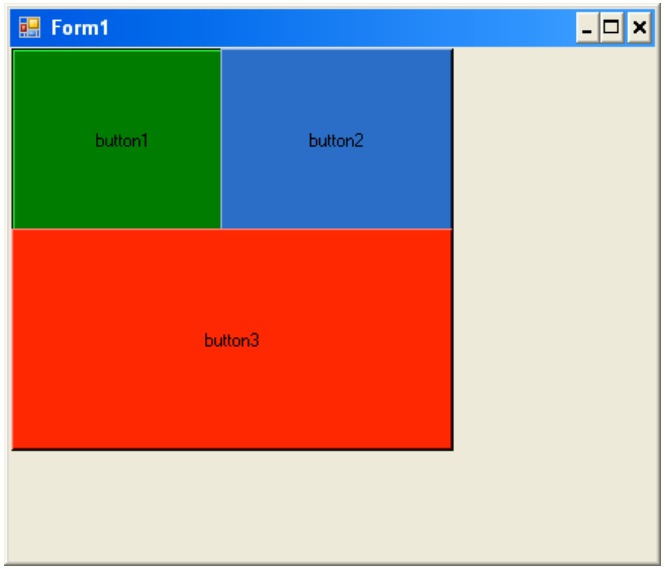
Overview 1st Topic: Auckland Layout Model

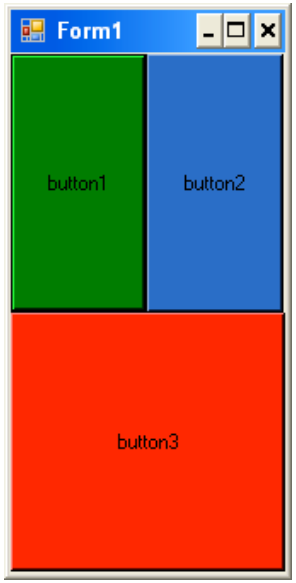
- GUI Layout
- The Auckland Layout Model (ALM)
- Abstraction, Modularity, Reuse
- Examples
- Document Orientation

Layout Managers

- Get a layout specification as input
- Recalculate the positions and sizes of the controls after each resizing



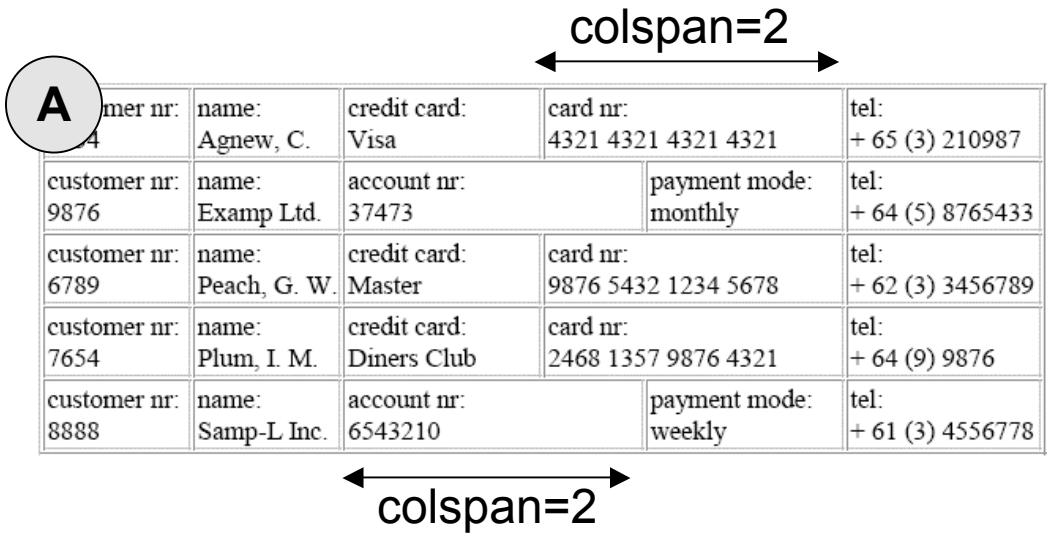


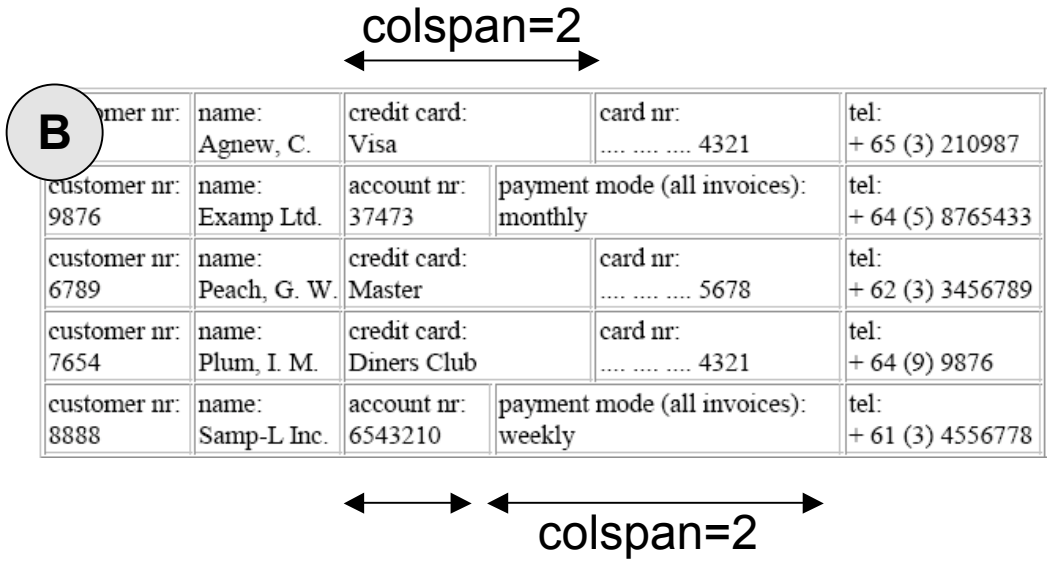


User Interface Layout with Ordinal and Linear Constraints [Lutteroth, Weber 2006]

customer nr: 1234	name: Agnew, C.	credit card: Visa	card nr: 4321 4321 4321 4321	tel: + 65 (3) 210987
customer nr: 9876	name: Examp Ltd.	account nr: 37473	payment mode: monthly	tel: + 64 (5) 8765433
customer nr: 6789	name: Peach, G. W.	credit card: Master	card nr: 9876 5432 1234 5678	tel: + 62 (3) 3456789
customer nr: 7654	name: Plum, I. M.	credit card: Diners Club	card nr: 2468 1357 9876 4321	tel: + 64 (9) 9876
customer nr: 8888	name: Samp-L Inc.	account nr: 6543210	payment mode: weekly	tel: + 61 (3) 4556778

Shortcomings of current approaches





Problem 1:

- Implementation hard-codes order of tabs
- Either A or B
- No free auto-adjustment possible
- Width of adjacent subtype-specific columns should be adjusted independently for each subtype

Shortcomings of current approaches

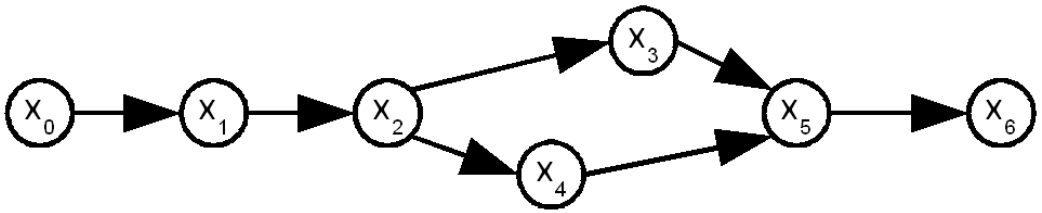
Problem 2:

A

mer nr:	name: Agnew, C.	credit card: Visa	card nr: 4321 4321 4321 4321	tel: + 65 (3) 210987
customer nr: 9876	name: Examp Ltd.	account nr: 37473	payment mode: monthly	tel: + 64 (5) 8765433
customer nr: 6789	name: Peach, G. W.	credit card: Master	card nr: 9876 5432 1234 5678	tel: + 62 (3) 3456789
customer nr: 7654	name: Plum, I. M.	credit card: Diners Club	card nr: 2468 1357 9876 4321	tel: + 64 (9) 9876
customer nr: 8888	name: Samp-L Inc.	account nr: 6543210	payment mode: weekly	tel: + 61 (3) 4556778

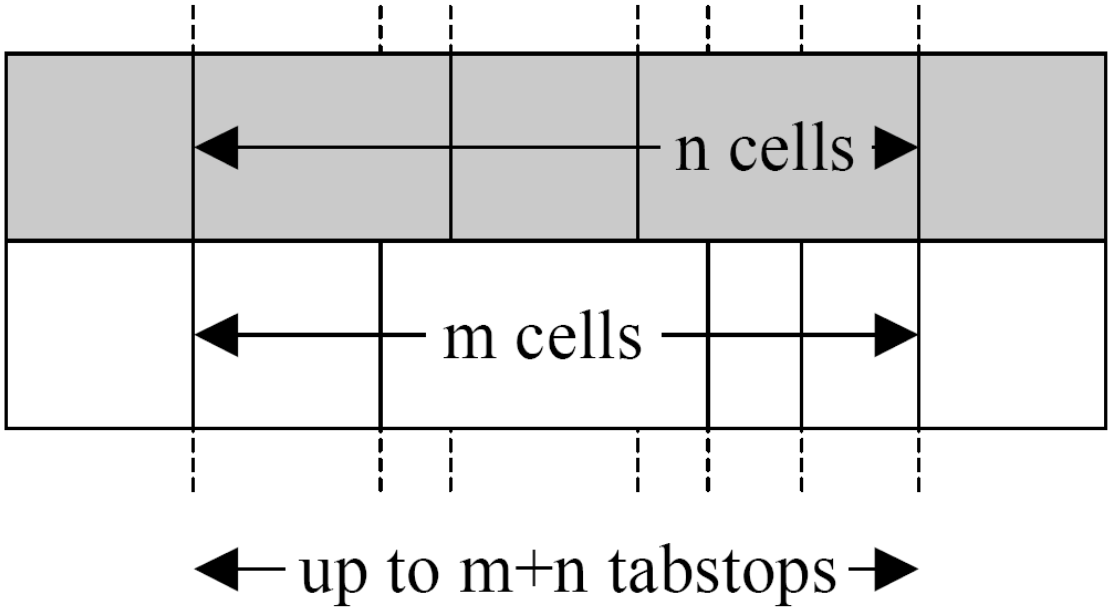
B

mer nr:	name: Agnew, C.	credit card: Visa	card nr: 4321	tel: + 65 (3) 210987
customer nr: 9876	name: Examp Ltd.	account nr: 37473	payment mode (all invoices): monthly	tel: + 64 (5) 8765433
customer nr: 6789	name: Peach, G. W.	credit card: Master	card nr: 5678	tel: + 62 (3) 3456789
customer nr: 7654	name: Plum, I. M.	credit card: Diners Club	card nr: 4321	tel: + 64 (9) 9876
customer nr: 8888	name: Samp-L Inc.	account nr: 6543210	payment mode (all invoices): weekly	tel: + 61 (3) 4556778



- Implementation of a subtype layout depends on implementation of other subtype layouts
  - Bad design:
  - Coupling!
- Need for partial order of column borders instead of total order

Larger-scale case



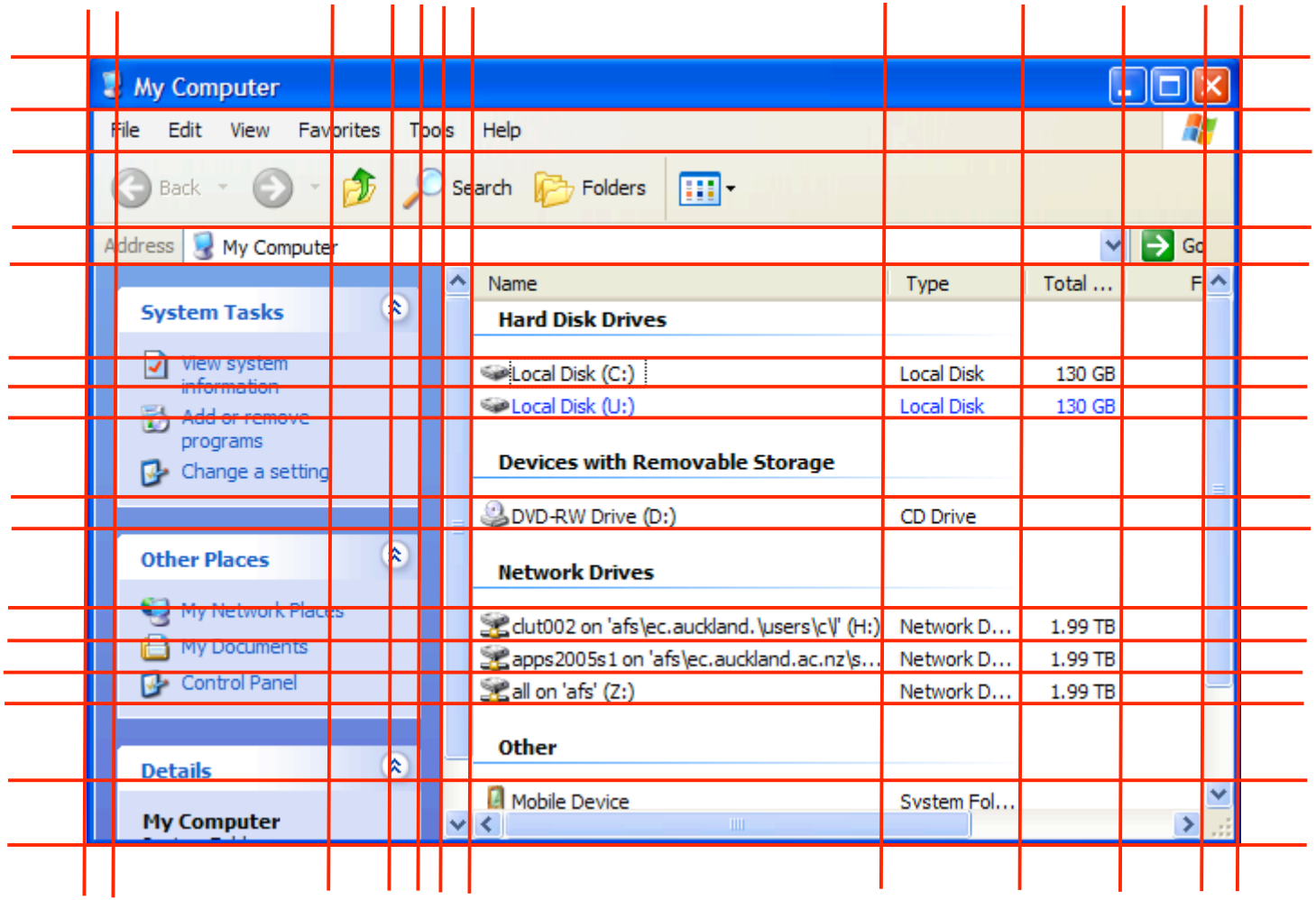
- In the first row: m has to be distributed over the n colspan attributes
- Result: very little adaptivity.

The Auckland Layout Manager

[Lutteroth, Strandh, Weber 2007]

- Focus switches from single table cells to borders of table elements:
- Vertical and horizontal tabulators,
- x-tabstops and y-tabstops (short: “tabs”)
- Each tabstop has a symbolic name
- Simply name the tabstops delimiting a particular cell instead of merging adjacent cells (“colspan” / “rowspan”)
- x-tabs and y-tabs are only partially ordered, respectively
- A table is a list of cell definitions, which create the partial orders of the tabs
- Each cell definition contributes one edge to the partial order of x-tabs and one edge to the partial order of y-tabs
- User does not have to fix the order of tabs if it is irrelevant, leaving more flexibility to the layout engine

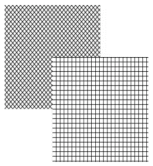
Specifying GUI Layout



Controls are aligned in a grid

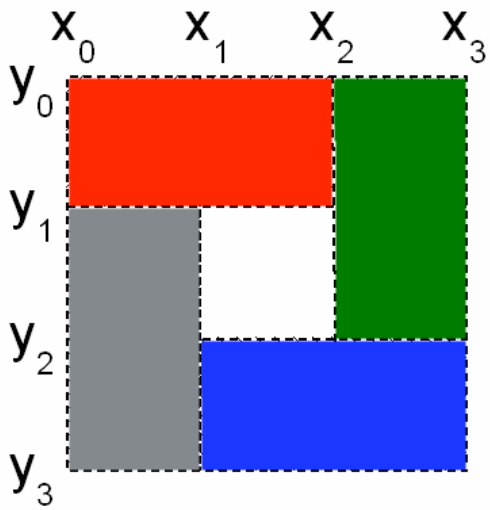
The Auckland Layout Model (ALM)

- Grid lines are variables with coordinates (*tabs*)
- Place controls by choosing left, top, right and bottom tab (*area*)
- Overlapping areas with layers

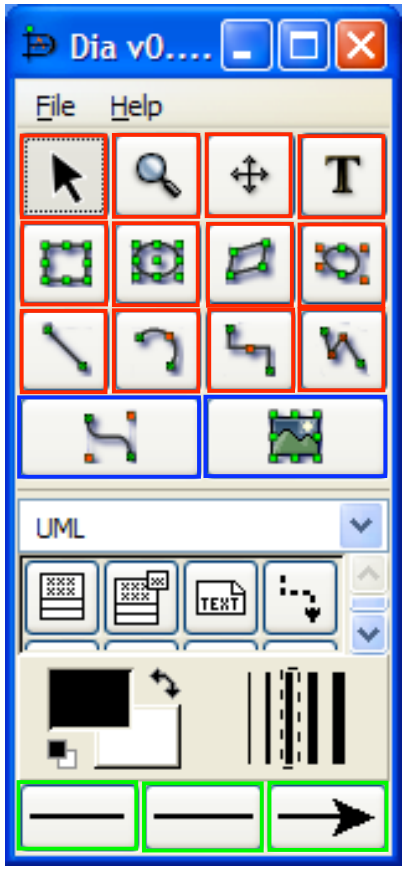


$$a =_{def} (x_1, y_1, x_2, y_2, layer, content)$$

$$\begin{aligned} A = \{&(x_0, y_0, x_2, y_1, 0, red), (x_2, y_0, x_3, y_2, 0, green), \\ &(x_1, y_2, x_3, y_3, 0, blue), (x_0, y_1, x_1, y_3, 0, grey), \\ &(x_1, y_1, x_2, y_2, 0, empty)\} \end{aligned}$$



Specifying GUI Layout

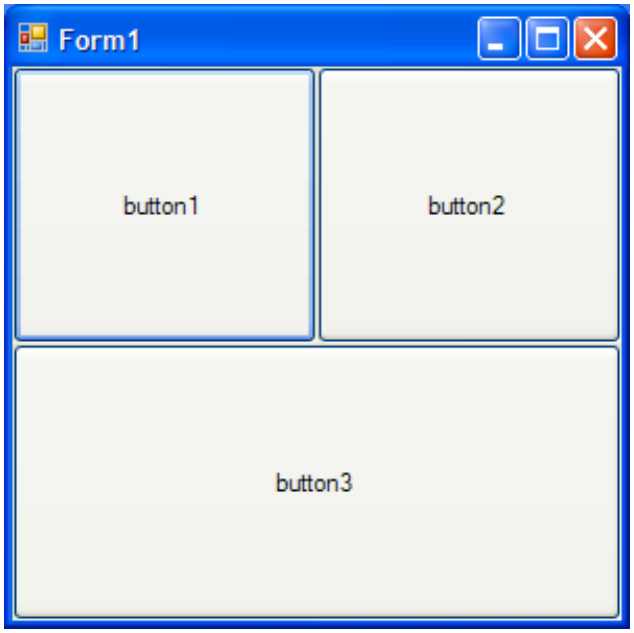


Same size

Same size
Same height as above
Double width as above

Same size
2/3 width as above

Example Layout




```
ALMEngine le = new ALMEngine();
public override LayoutEngine
LayoutEngine { get { return le; } }

{…
LayoutSpec ls = new LayoutSpec();
XTab x1 = ls.AddXTab();
YTab y1 = ls.AddYTab();

ls.AddArea(ls.Left, ls.Top, x1, y1, button1);
ls.AddArea(x1, ls.Top, ls.Right, y1, button2);
ls.AddArea(ls.Left, y1, ls.Right, ls.Bottom, button3);

ls.AddConstraint(2, x1, -1, ls.Right,
    OperatorType.EQ, 0);
ls.AddConstraint(2, y1, -1, ls.Bottom,
    OperatorType.EQ, 0);
…
}
```

Linear Constraints in the ALM

$$C \subset \{ \; a_0 x_0 + \ldots + a_m x_m + b_0 y_0 + \ldots + b_n y_n \; OP \; c$$
$$\mid \; a_0, \ldots, a_m, b_0, \ldots, b_n, c \in \mathbb{R} \wedge OP \in \{\leq, =, \geq\}\}$$

- Absolute constraints

  $$x_3 = 50.$$

- Relative constraints
  - Relative position $x_2 - x_1 = 100.$
  - Relative size

    $$x_2 - x_1 = x_3 - x_2 \Leftrightarrow -x_1 + 2x_2 - x_3 = 0.$$

    X_1—x_2—x_3

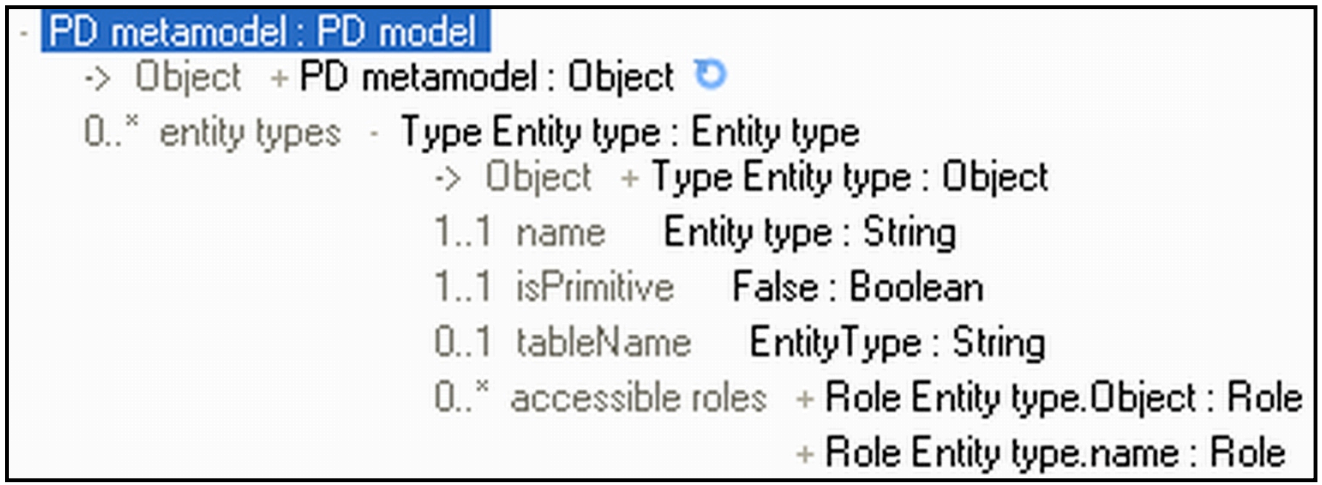
    $$x_2 - x_1 = 2(x_4 - x_3) \Leftrightarrow -x_1 + x_2 + 2x_3 - 2x_4 = 0.$$

  - Aspect ratio

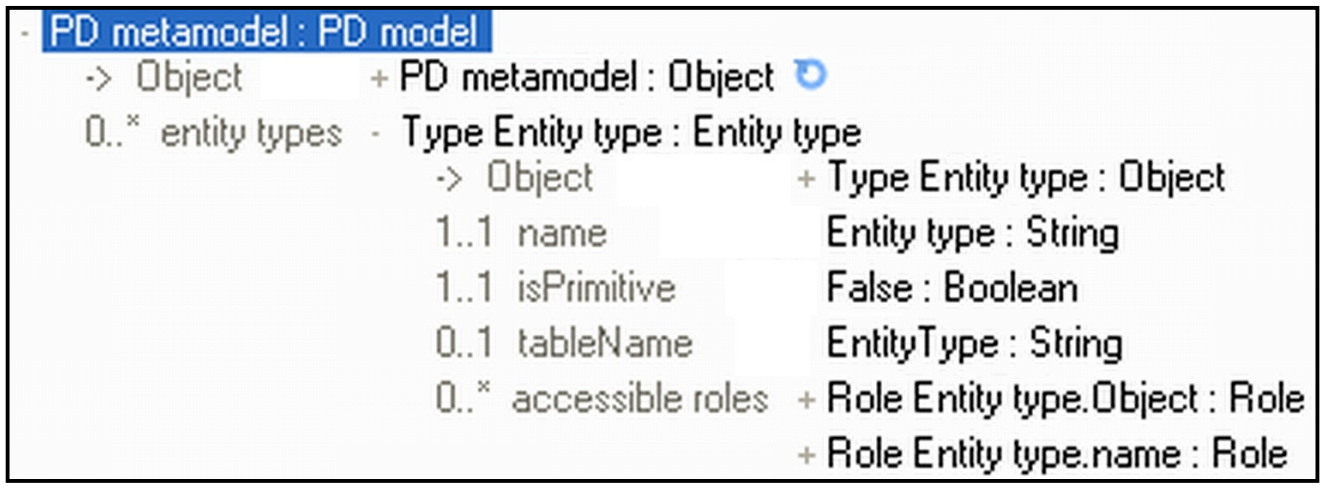
    $$\frac{x_2 - x_1}{y_2 - y_1} = \frac{16}{9} \Leftrightarrow -x_1 + x_2 + \frac{16}{9} y_1 - \frac{16}{9} y_2 = 0.$$

- Different units possible per constraint (cm, pixels)

Example: Tree View



Recursive Implementation



ALM Implementation