What's in a Relationship?

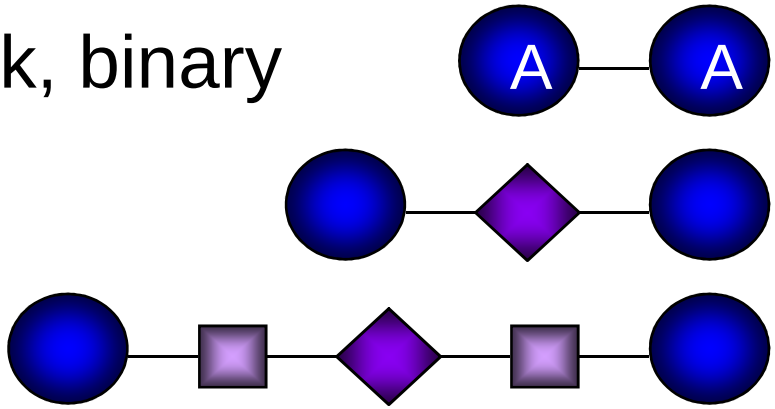
Distinguishing Property Holding and Object Binding

MetaCASE Background

- Modelling methods, not the real world
- MetaCASE useful, but needs extending
  - more powerful data model
  - support interlinked methods & models
- Need integrated metaCASE and CAME
  - support method component reuse
  - one data model for model and metamodel

Relationships: background

- Increase in relationship-like concepts
  - Relational, network, binary
  - ER
  - OPRR



- We added concepts…
  - …but we never said what the line was!

Object Binding, Property Holding

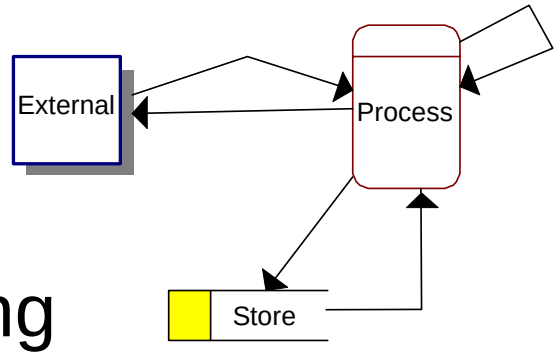
- A relationship binds objects together
  - e.g. ’marriage’ for man & wife
- A relationship has properties
  - e.g. ’date of marriage’
- Binding concerns objects & relationship
- Properties only concern the relationship

Relationship Problems (I)

- Reduces similarities between metatypes
- Complicated handling of binding actions
  - no one concept knows enough
- Binding information duplicated
- Prevents type reuse

Relationship Problems (II)

- Lower expressive power
  - cannot model relationship types with more than one binding

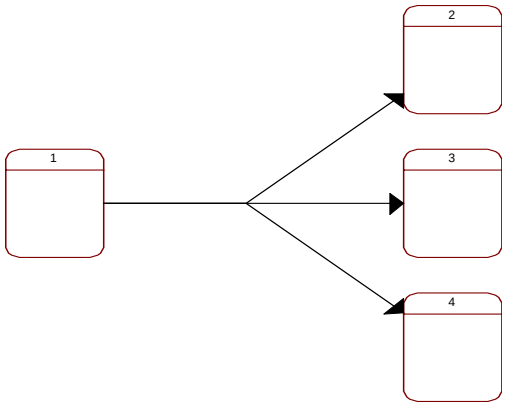


- Inefficient for multi-user applications
  - complicated links, duplicated information

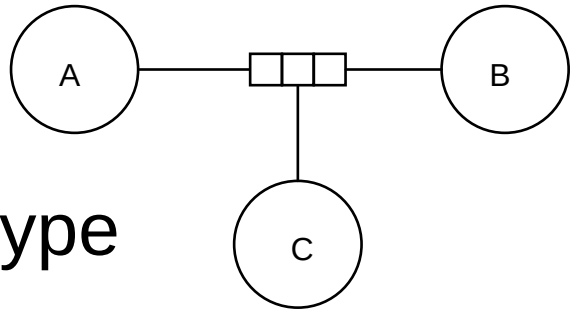
    $\Rightarrow$ more locking, low concurrency

More Roles and Objects

- Many objects per role
  - e.g. some DFDs
  - …and most metamodels

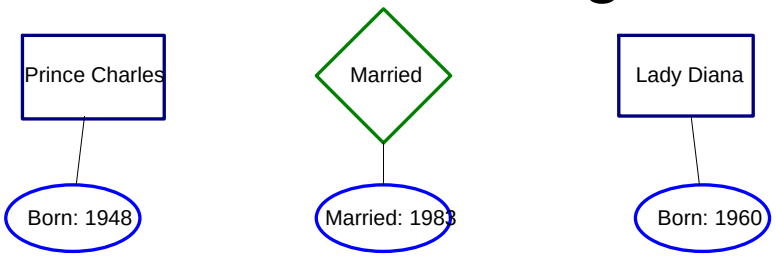


- Many roles per relationship
  - e.g. NIAM / ORM
  - new role types vs.
    many occurrences of same type

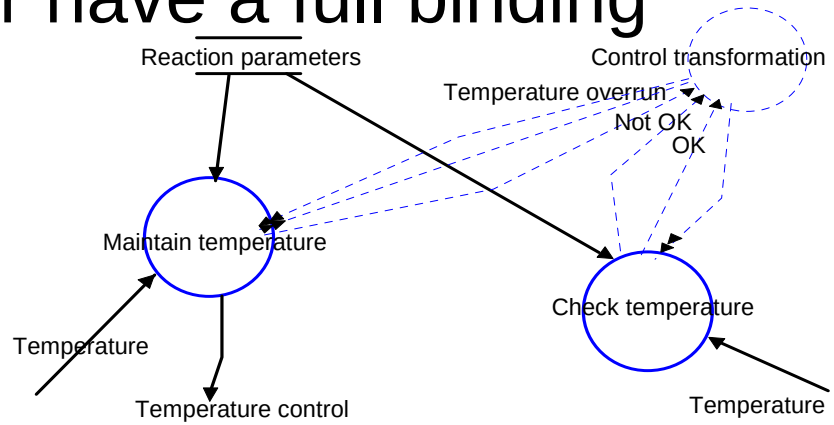


Relationship Properties Independent of Binding

- Relationship can be drawn before binding
  - e.g. many ER tools
  - binding added later
  - similar to empty graph

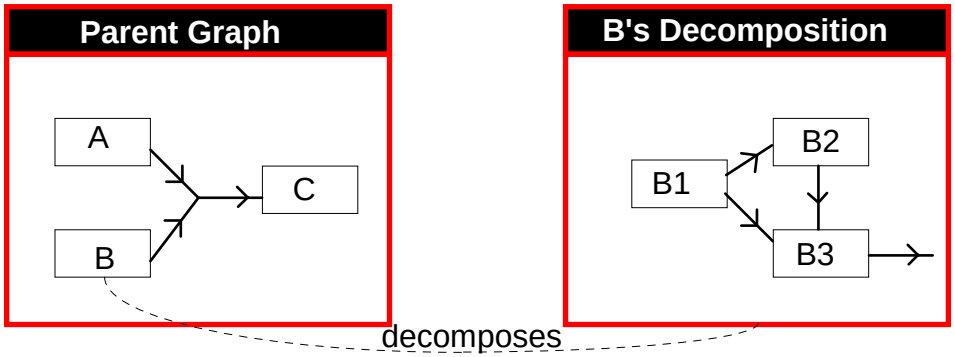


- Relationship might never have a full binding
  - e.g. triggers in RTSA
  - how to model interface relationships?



Role Independent of Relationship

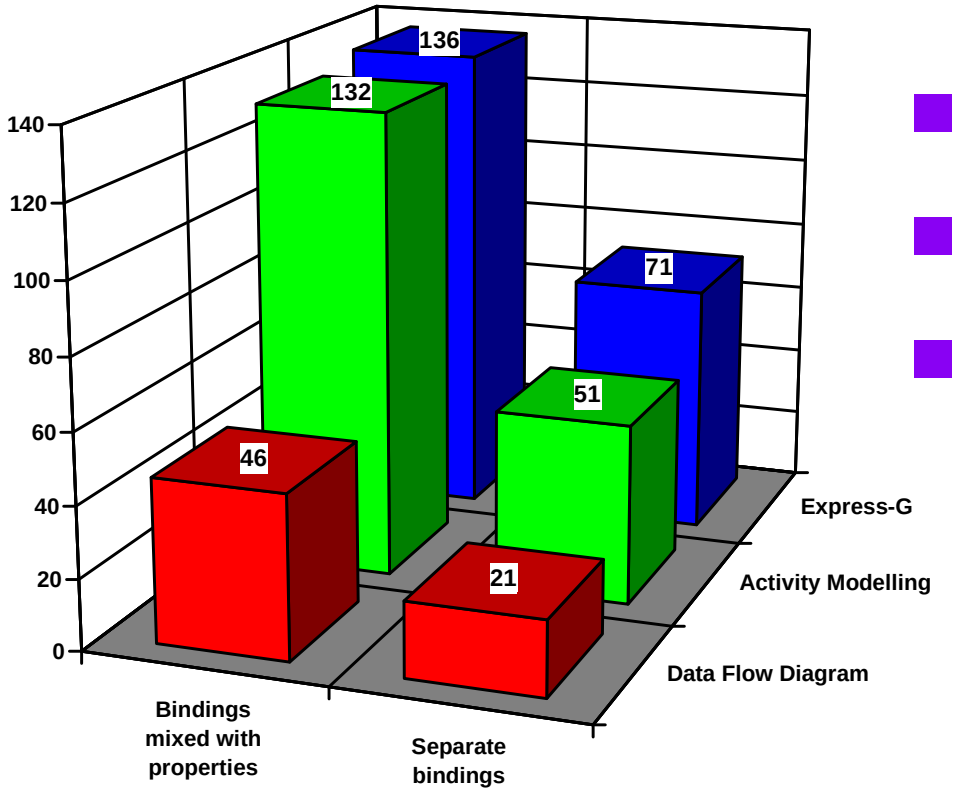
- Lemma 1: a line is a role
- Lemma 2: a junction is a relationship
- Decomposition interface lines are roles without relationships, e.g. line from B3



Binding Structure

- Binding $\rightarrow$ Relationship Role&Objs^{+}
  - We may have n-ary relationships
  - Sometimes relationship part may be empty, e.g. interface bindings
- Role&Objs $\rightarrow$ Role Object^{+}
  - Many objects may be attached to a role

Implementation Efficiency



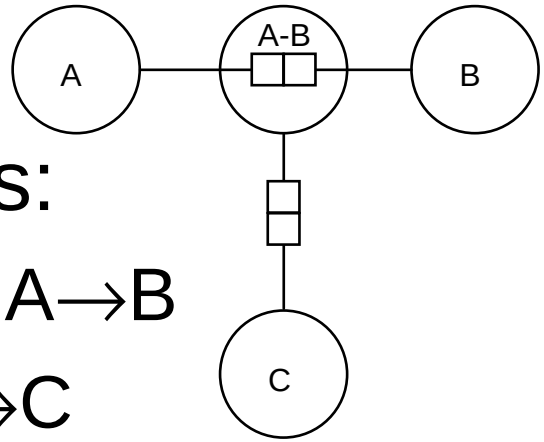
- Storage 2:1
- Speed 2:1
- Locking 3:1 - 4:1

Graphs and Bindings

- Bindings contained within a graph
  - same relationship has different bindings in different graphs $\Rightarrow$ reuse
- Graph 'knows' its contents
  - so can know facts involving several of them
- Graph $\rightarrow$ object* role* rel.* binding*
- Partial bindings for interface relationship

Polymorphism of Meta-Types

- Some relationships behave like objects
  - e.g. in NIAM
- We can model this with bindings:
  - A-B in relationship slot in binding A→B
  - A-B in object slot in binding A-B→C
- A-B thus has dual nature: polymorphism
- Also applicable on type level



Conclusions

- Conceptual improvements
  - Object, role, relationship more similar
  - N-ary relationships, multiple roles per object
  - Polymorphism of metatypes
  - Ability to integrate methods and models
- Efficiency
  - storage, speed, multi-user
- Reuse of types