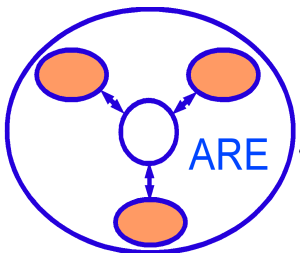
Automatic Roundtrip Engineering

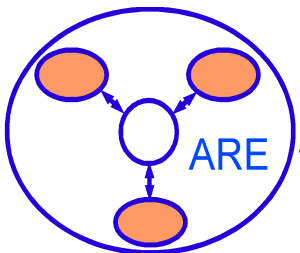
Uwe Aßmann



Why Compilers Will Live Forever

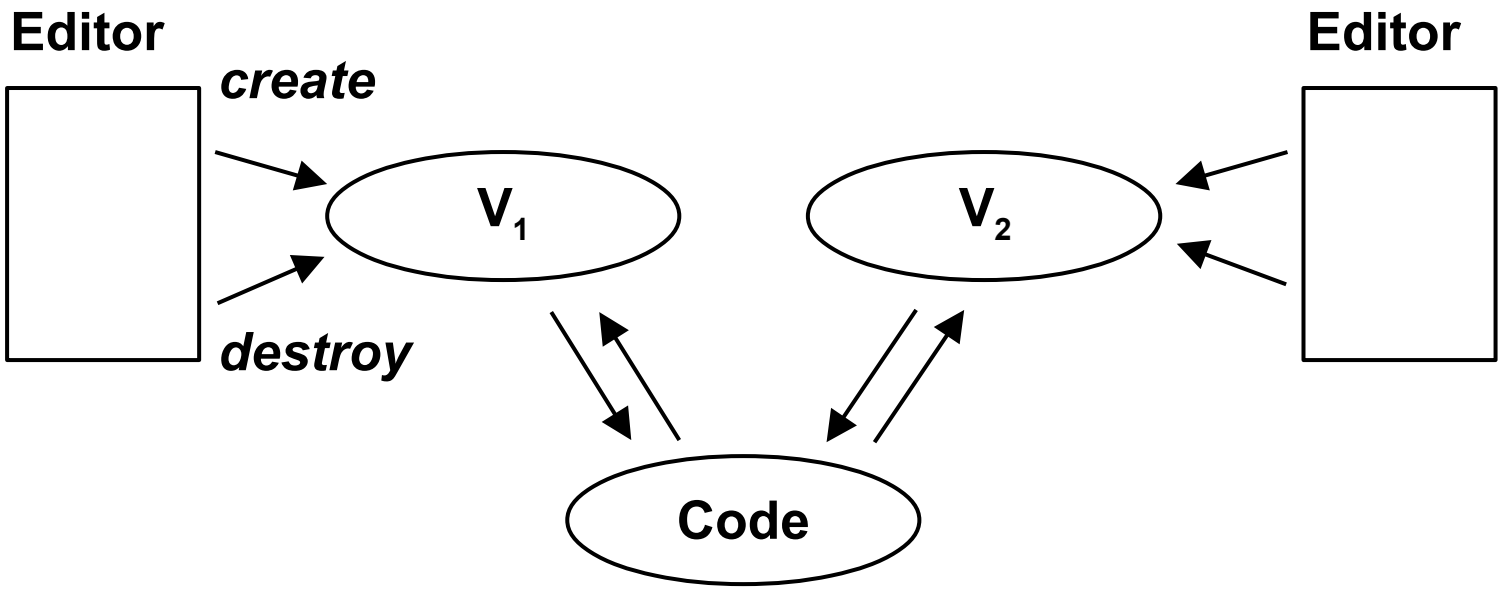
And why you should prepare to get more funding in the future....

RISE

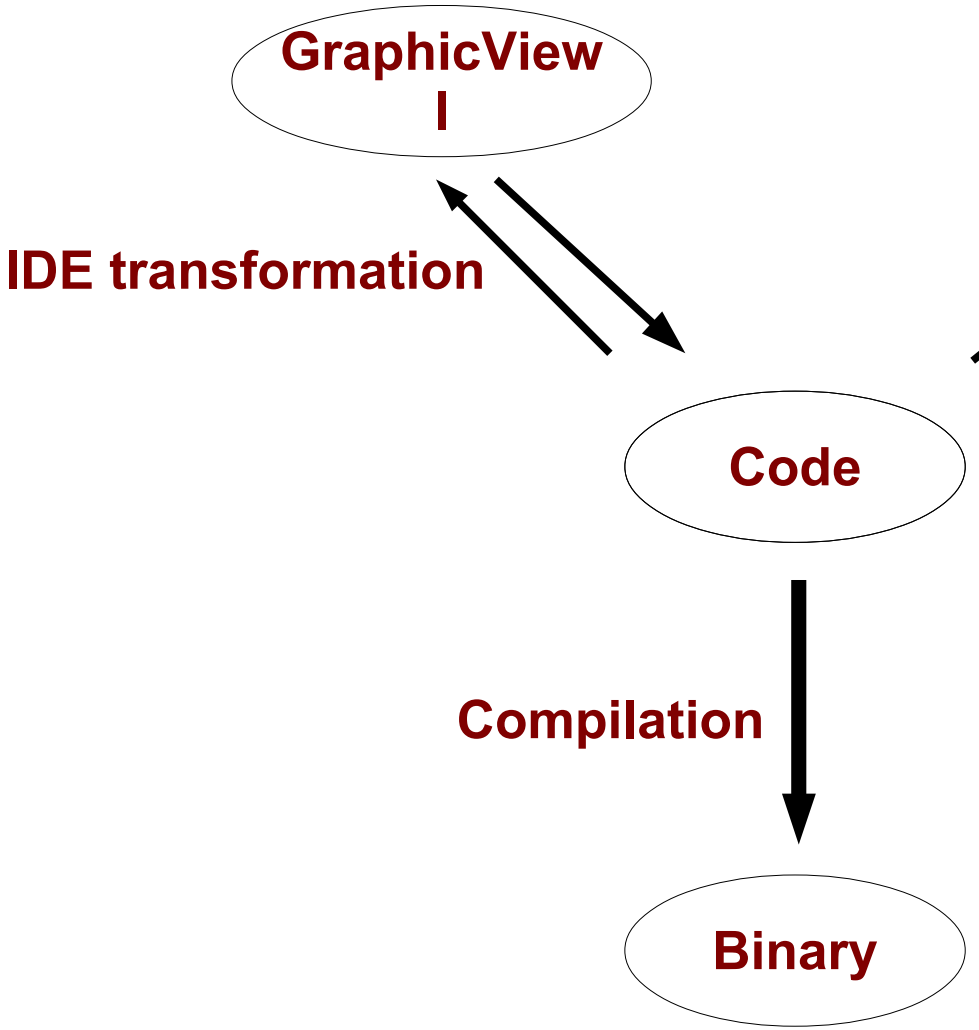


easycomp

1) Modern Case Tools Support Roundtrip Engineering

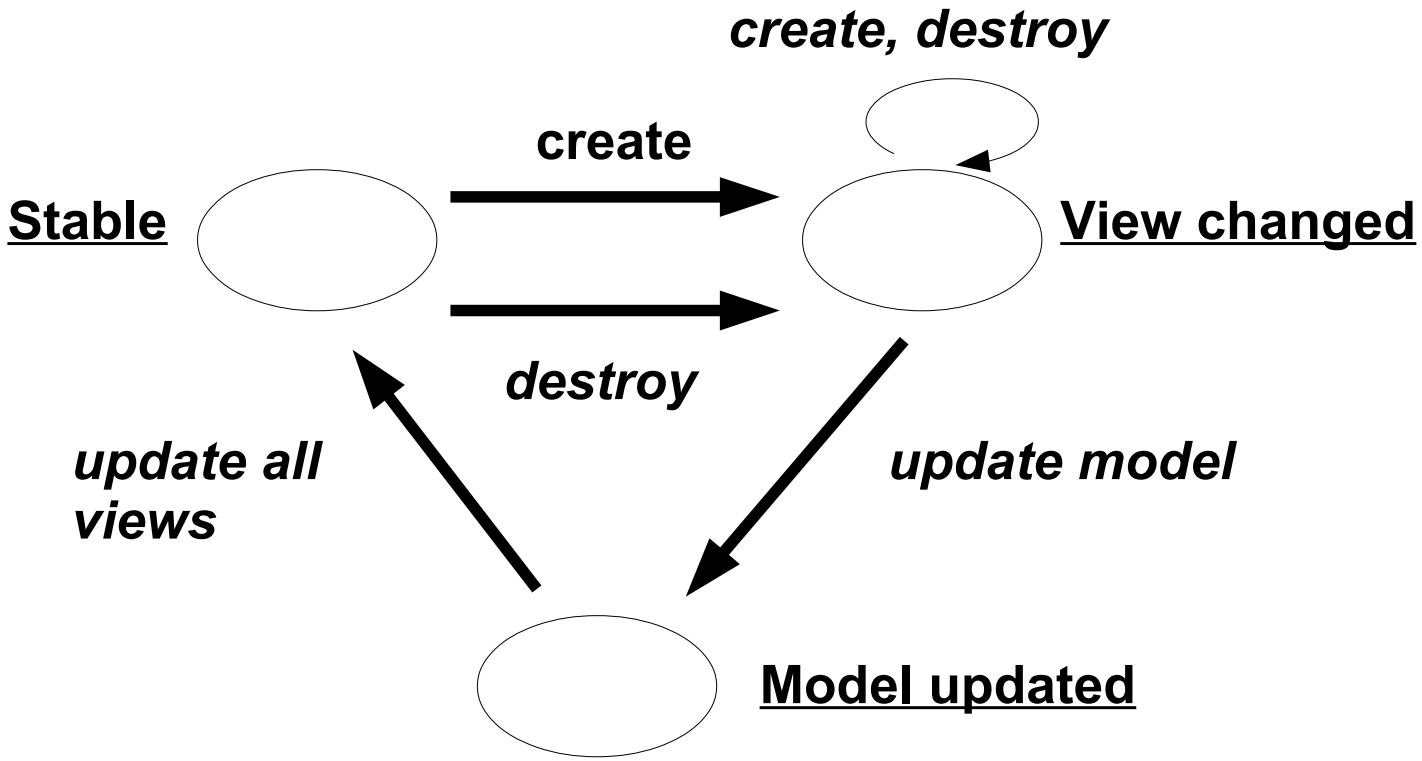


IDE Adds RE-Preprocessing To Compilation



How to Systematize?
How to implement it not with hacking?
How to add more views without problems?
How to formalize the problem?

Roundtrip States and Events



2) The Batch System Challenge

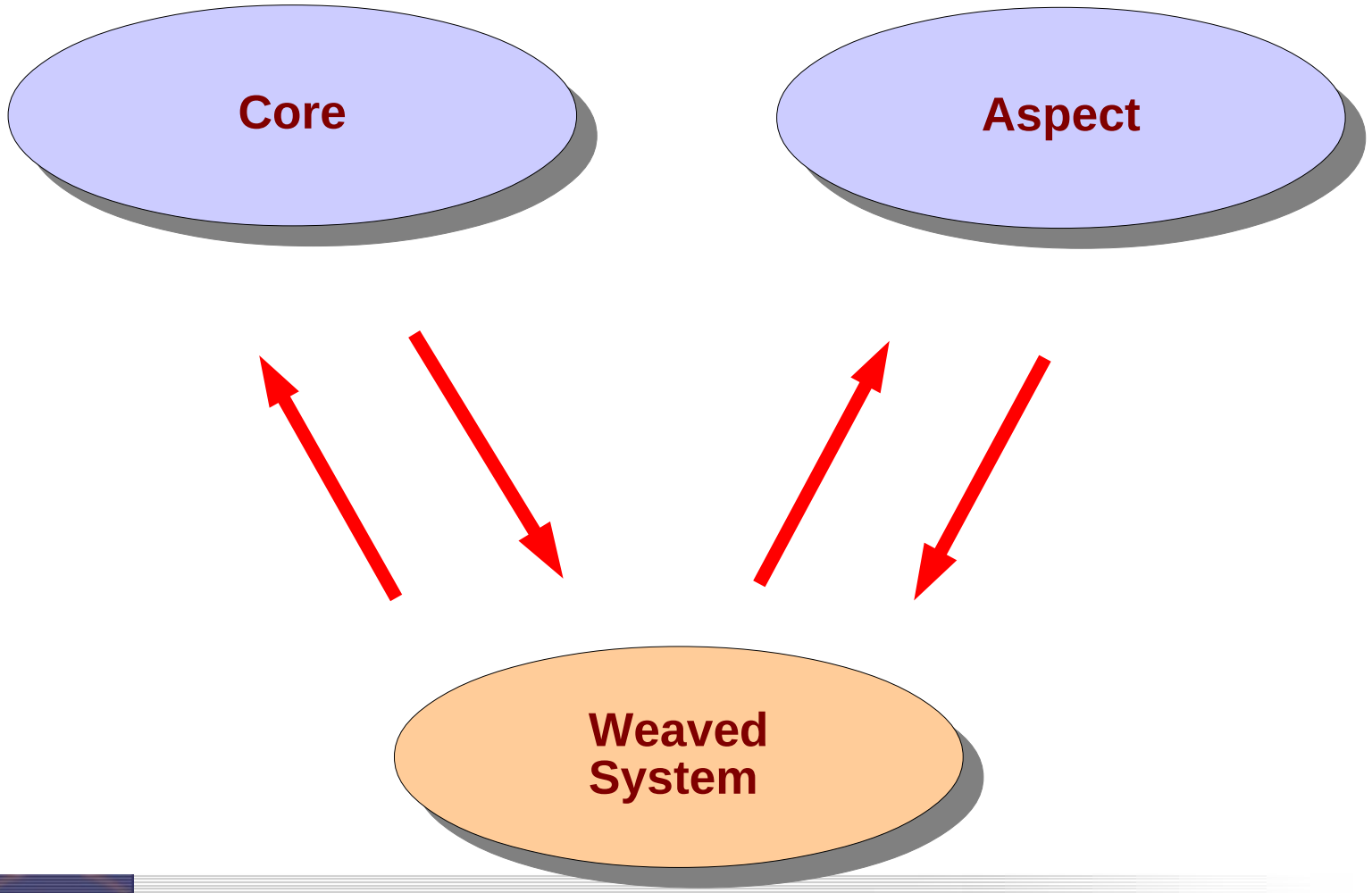
How to Unify TeX and Emacs?

How to unify batch systems with

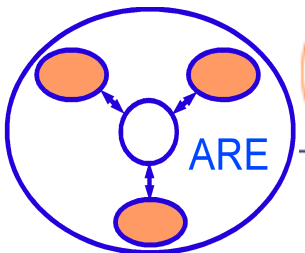
- Incremental processing
- Editing the results and tracing it back to the input

How can we edit the bitmap, tracing it back to TeX?

3) How to Debug AOP Systems?



Domain Transformations and Roundtrip Engineering



Roundtrip Engineering (RE) in Mathematics and Algorithmics

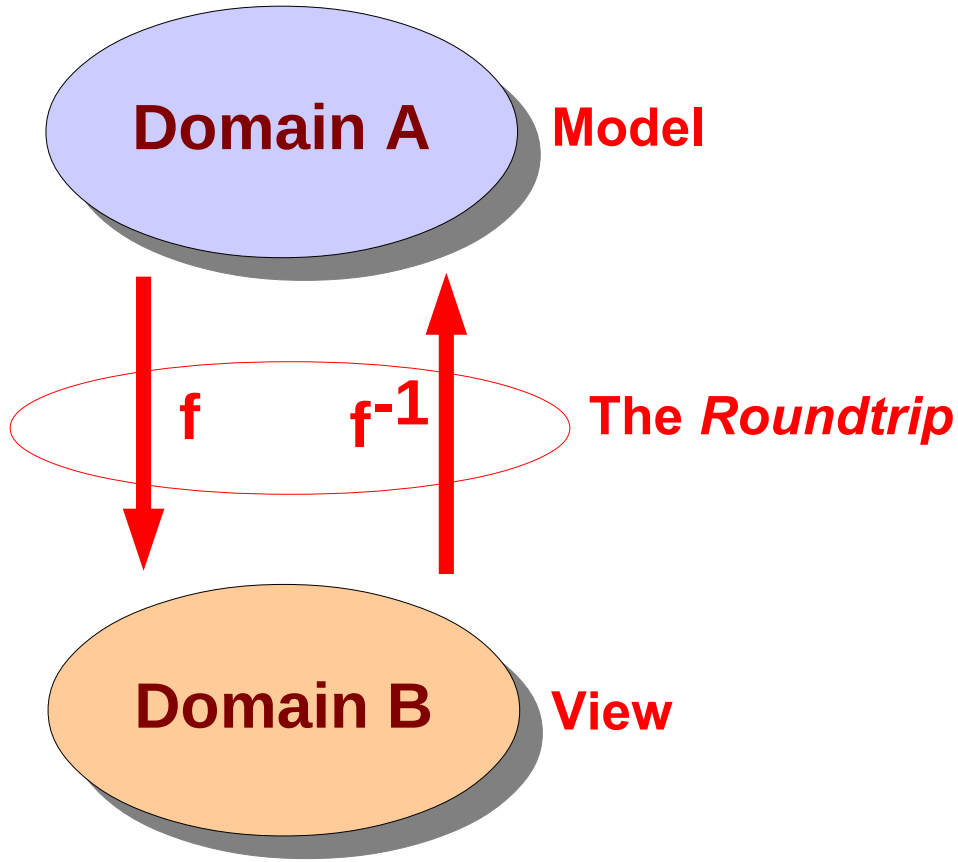
Mathematics and Algorithmics know the problem solving principle of **domain transformations (roundtrip)**

If a problem is too hard to solve in a domain D,

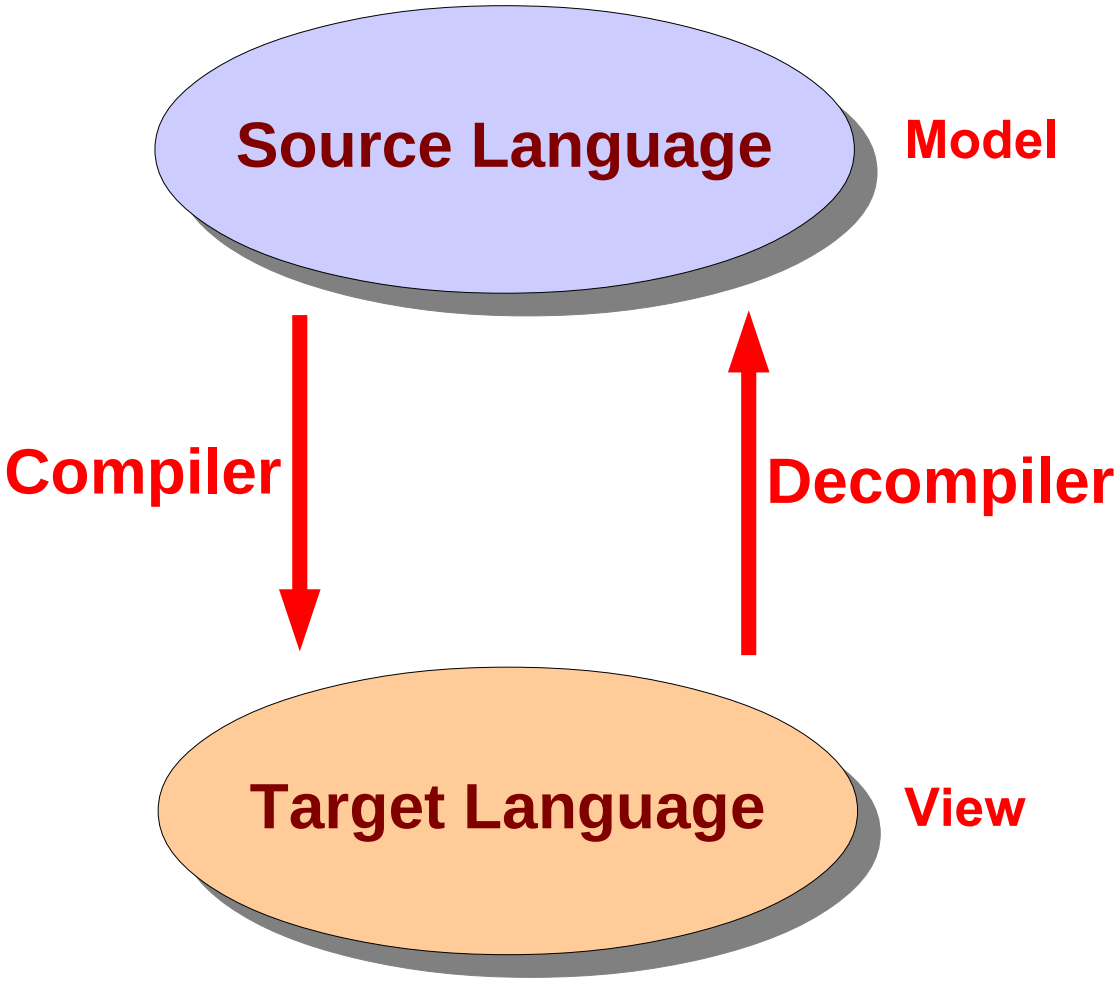
- transfer it into another domain D'
- solve it with a simpler algorithm
- transform it back with the inverse domain transformation

Examples: Fourier, Wavelet, Laplace transformation

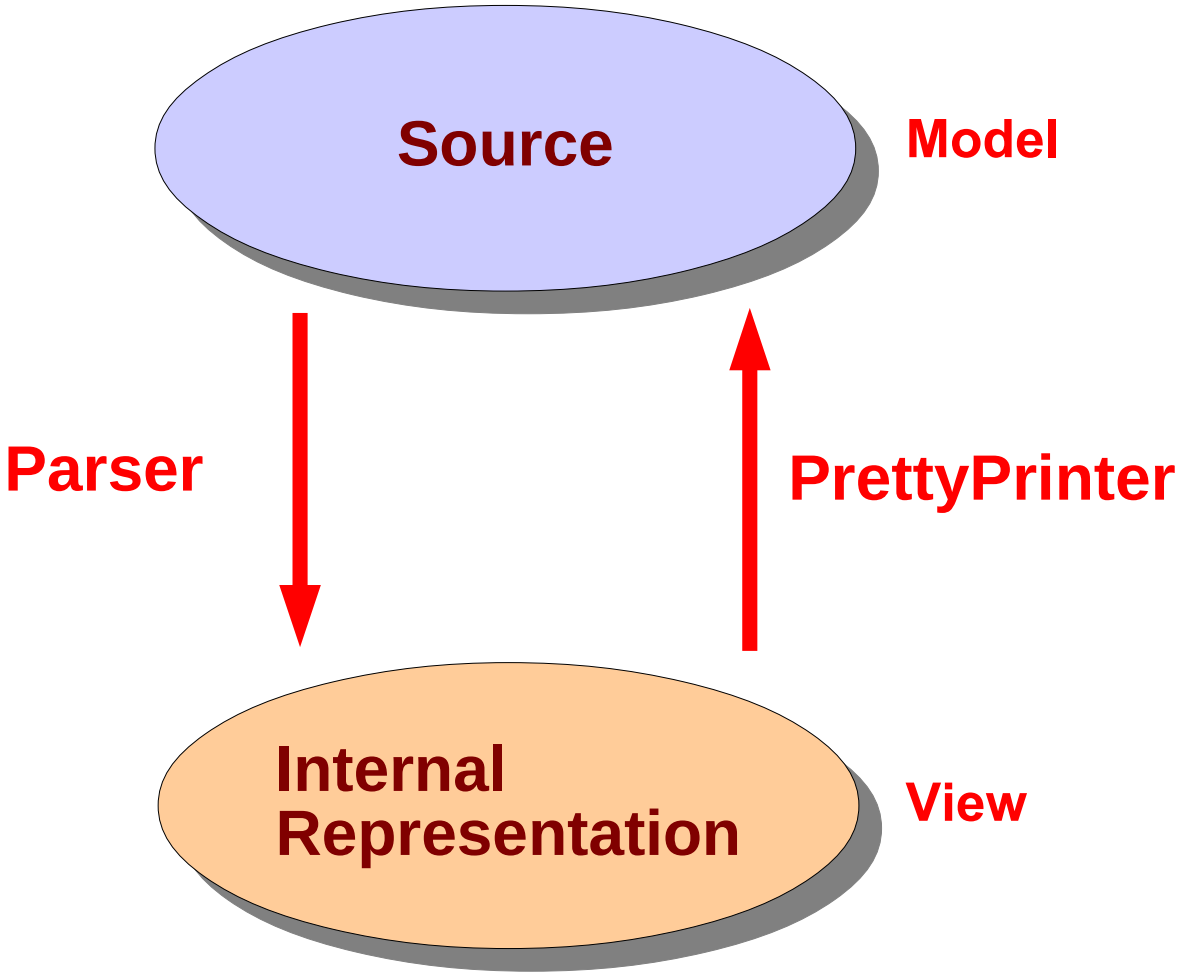
We call D the **model** and D' the **view**



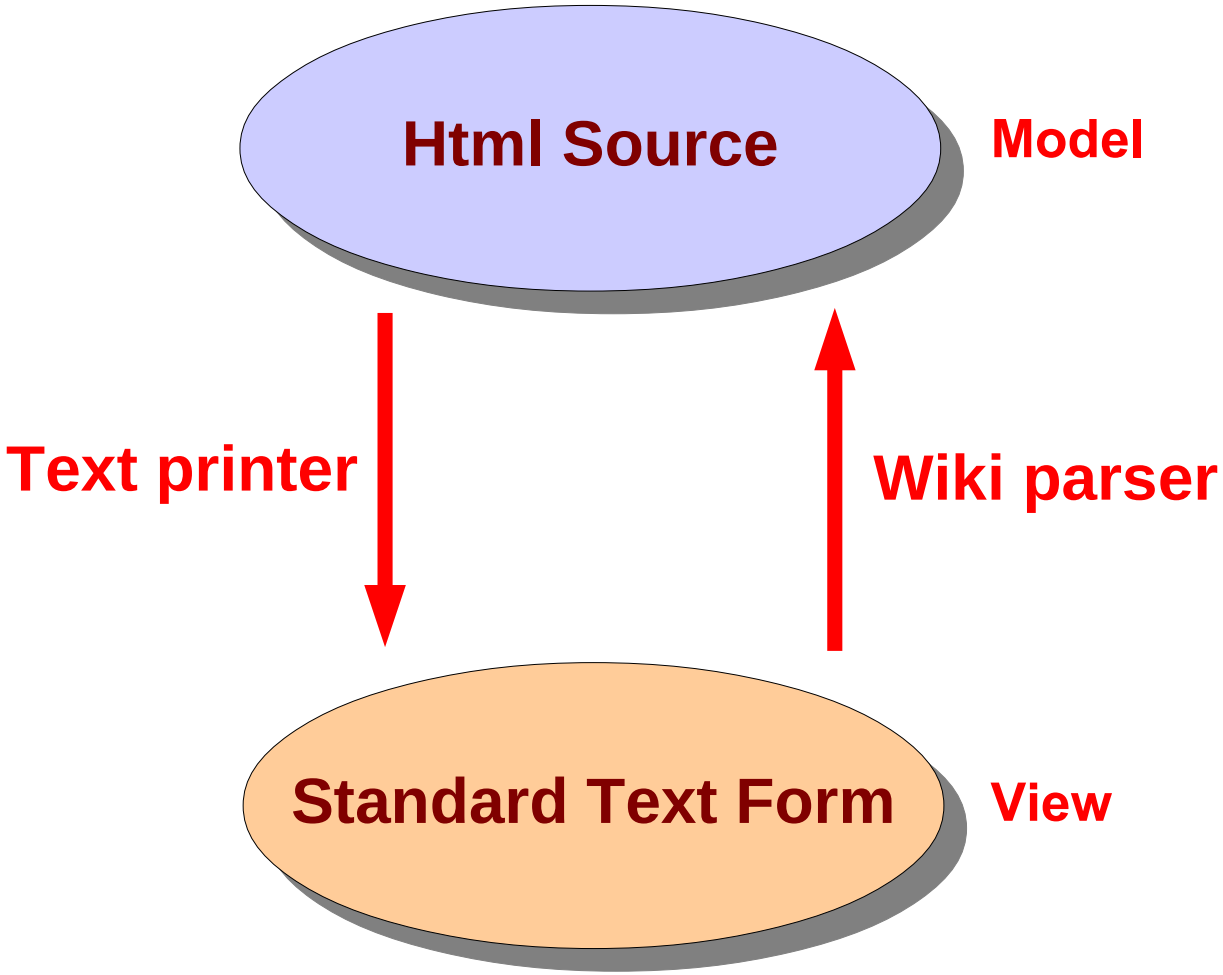
Domain Transformations in Compilers



Refactoring Tools (e.g., sf.recoder.net)



Wikis


Html Source
Model
Text printer
Wiki parser
Standard Text Form
View

Other Examples

Wikis are domain transformations between html and text

- Special markup recognizes semantics, i.e., links

XML and NiceXML

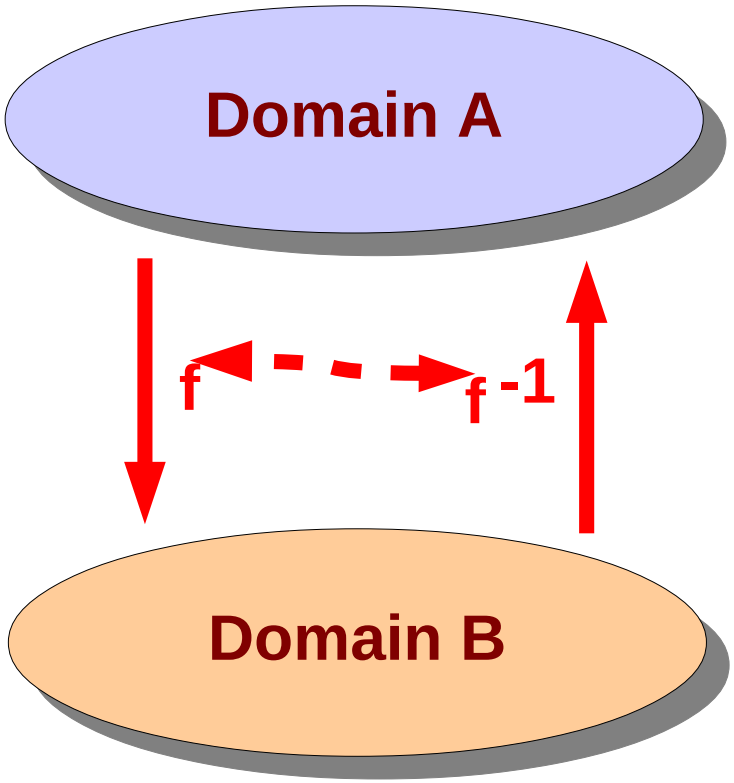
RDF and N3

Automatic Roundtrip Engineering

Automatic Roundtrip Engineering (ARE)

If the inverse domain transformation can be automatically computed, we speak of **Automatic Roundtrip Engineering**

Inverse domain transformation is for free



ARE - The Definition

Let A, B be two domains
f:A->B a domain transformation from a function space F
i:F->F a functional that calculates an inverse, the **inverse generator**

R = (A, B, f, i) is an automatic roundtrip system (ARE)

(f, i(f)) is a Galois connection, in which the inverse can be computed automatically

What is ARE?

An architectural style

- Working together on an artefact in different views
- Only half of the transformations need to be specified

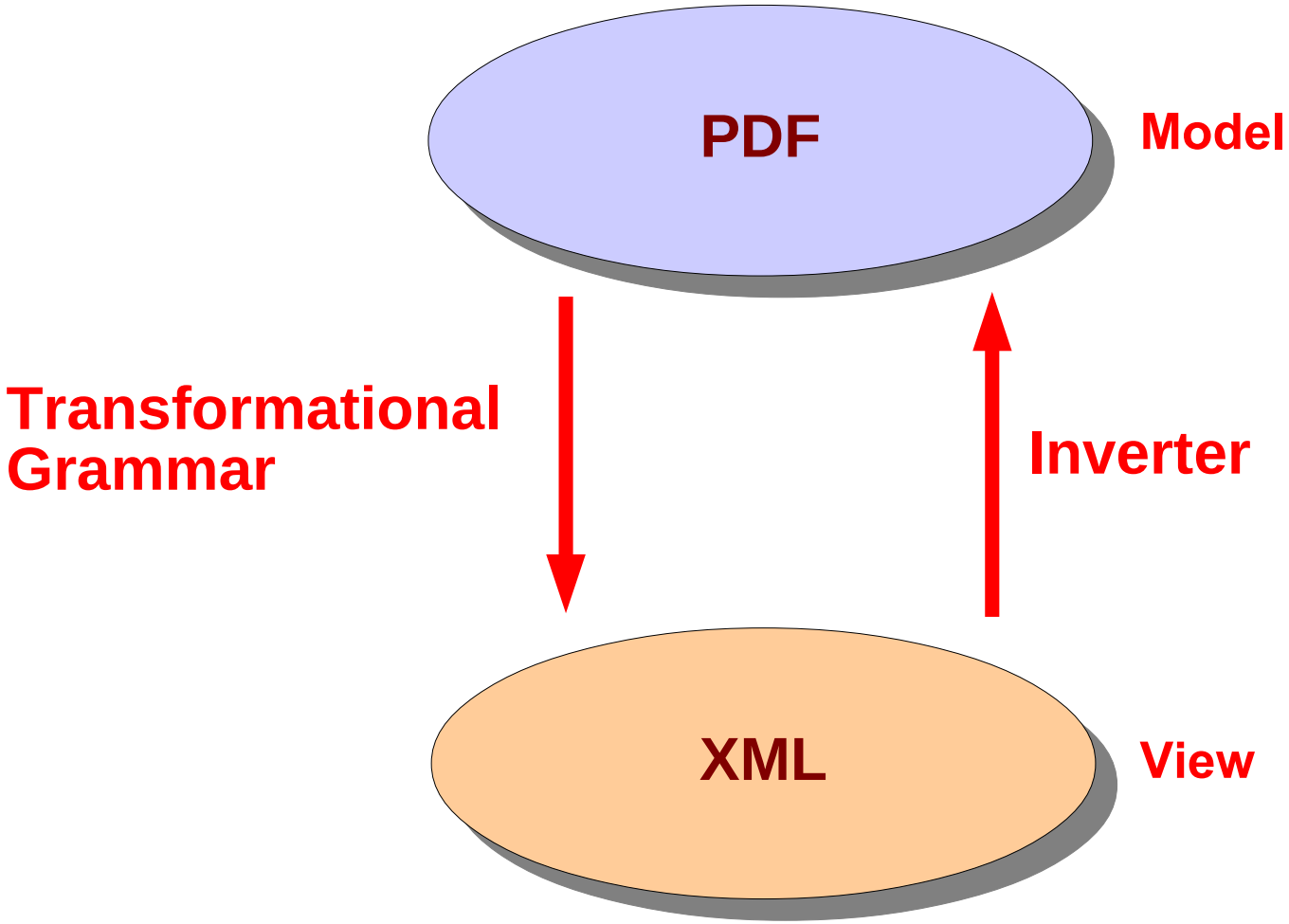
A design pattern

- Resembling MVC (model view controller), but with one view only

An algorithm class

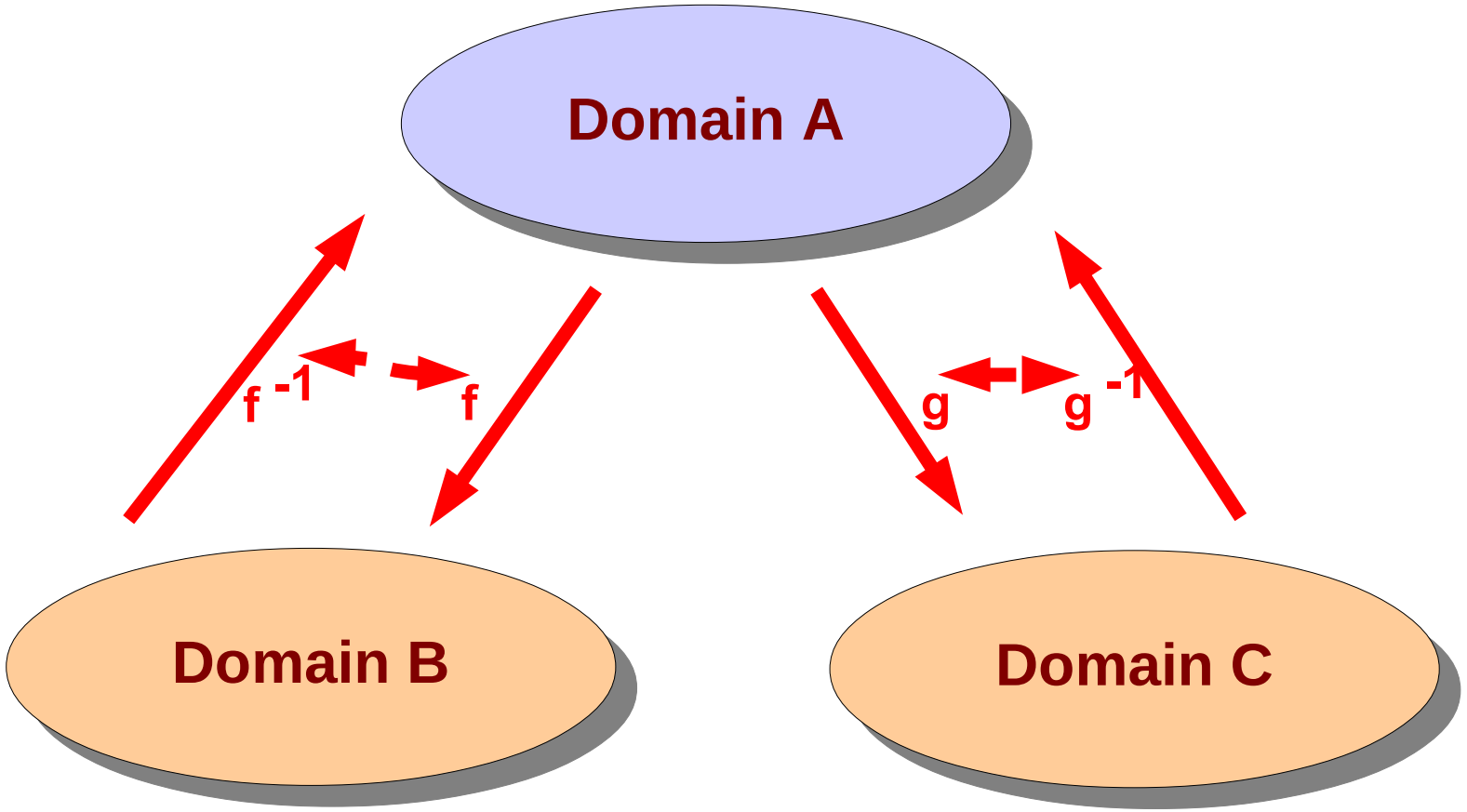
- A subclass of domain transformations

Adobe XPDF: PDF <-> XML



Transformational Grammar should be invertible

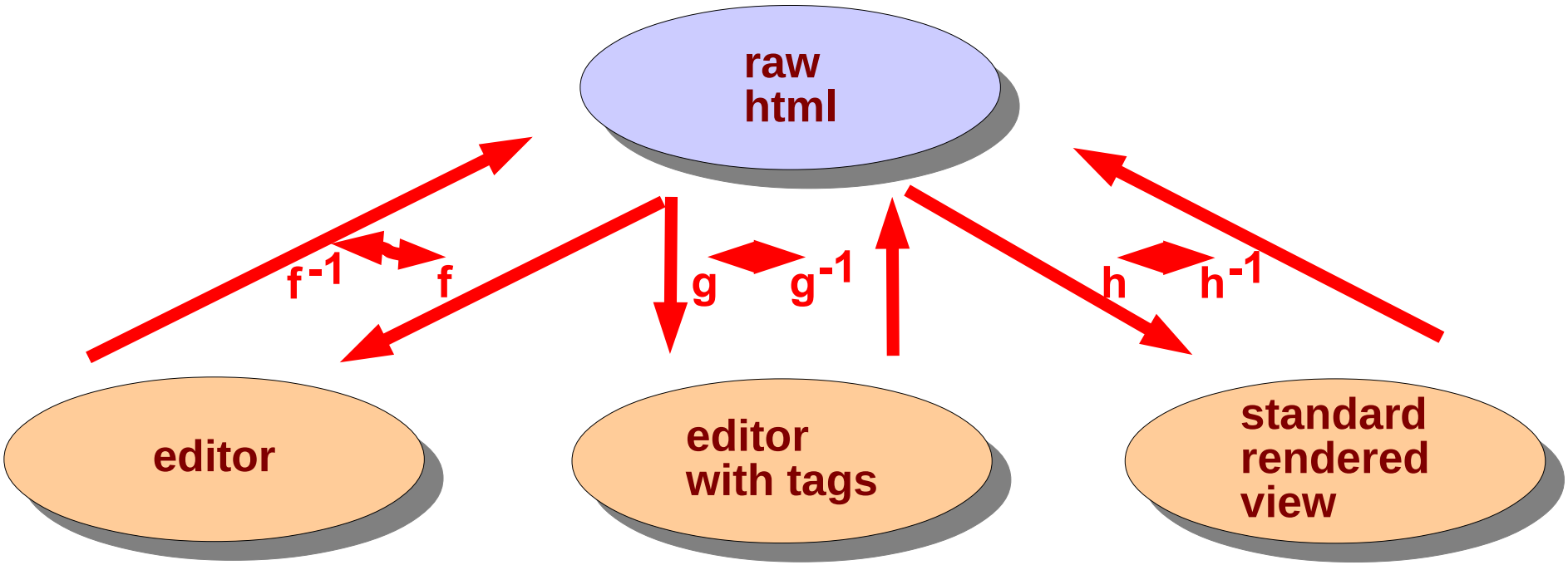
ARE with Multiple Domain Transformations



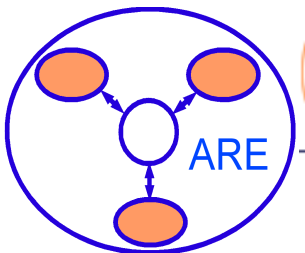
Example Mozilla Composer

The mozilla composer is an html editor with 1 model and 2 views

the views raw html, edit, edit-with-tags can be editied, the standard rendered view is view-only



Model-View based Automatic Roundtrip Engineering (MVARE)



easycomp

MVARE with Projections and Integrations

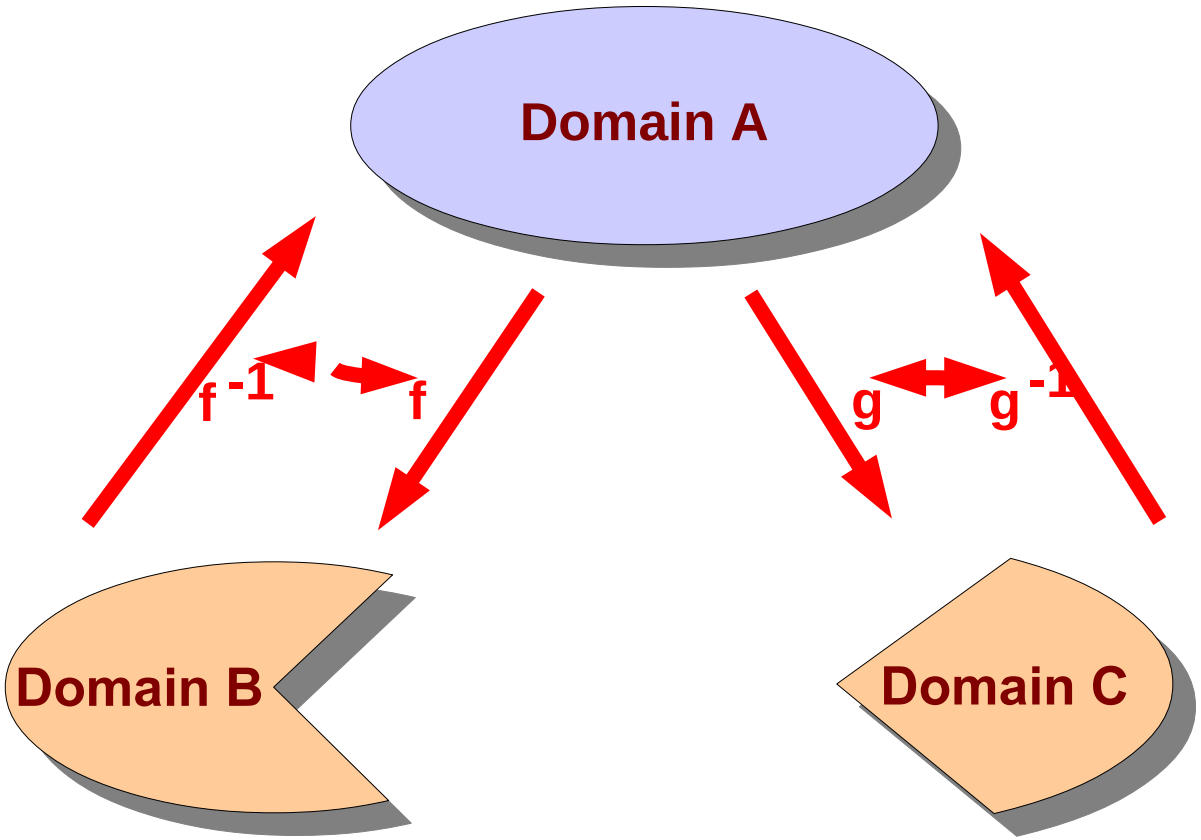
View (IEEE):

> "A form of **abstraction** achieved using a selected set of architectural concepts and structuring rules, to focus on particular concerns within a system"

A **Model-View ARE (MVARE)** is an ARE with multiple domain transformations that

- decompose a domain (**projections**),
  - domain transformationst that project to a simpler domain with less information
  - i.e., from inverting one domain transformation not the entire model can be recompouted
- And their inverses, that **integrate** the domains again (**integrations)**

MVARE with Projections



A view can be substantially simpler than the model

There may be solution algorithms in a view which are much more efficient than any algorithm on the model

The projections **decompose** the model into simpler items

While the integrations **compose** the simpler solutions into a main solution

MVARE – The Definition

Let A, B1, .., Bn be n+1 domains
f[j]:A->Bj projecting domain transformations from a function space F
i:F->F a functional that calculates an inverse, the **inverse generator**

Then, R = (A, B, f[1],..,f[n], i) is an **model view automatic roundtrip system (MVARE)**

A is called the **model**
B1,..,Bn are called the **views**

MVARE Applies Divide and Conquer

A view can be substantially simpler than the model

There may be solution algorithms in a view

- which are much more efficient than any algorithm on the model

The domain transformations decompose the model into simpler items

- While the inverse domain transformations compose the simpler solutions into a main solution

MVARE Examples

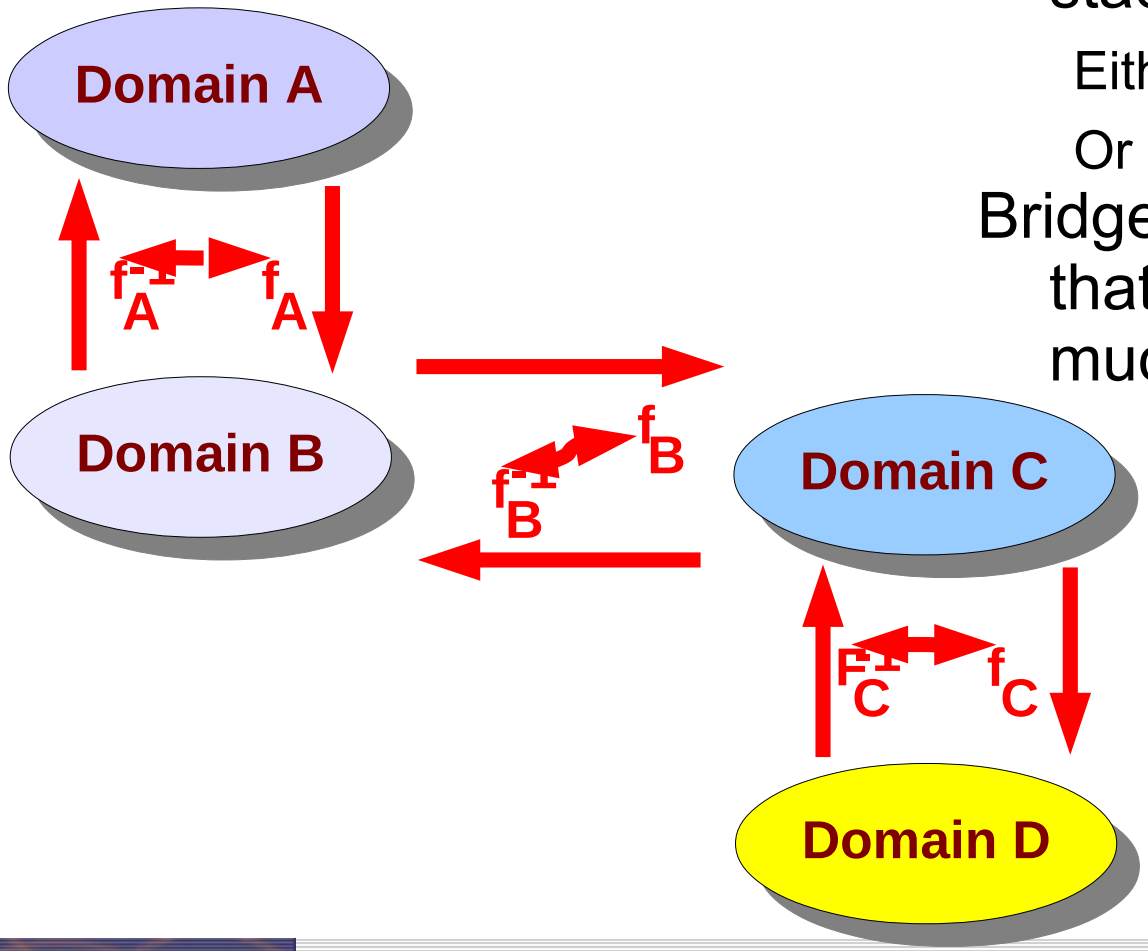
Executable UML Tools

- Together

SGML (separation of structure and layout)

CODEX is an MVARE with DPO graph rewriting and memoization of complete redexes [Larsson 2002]

ARE Chains (CARE)

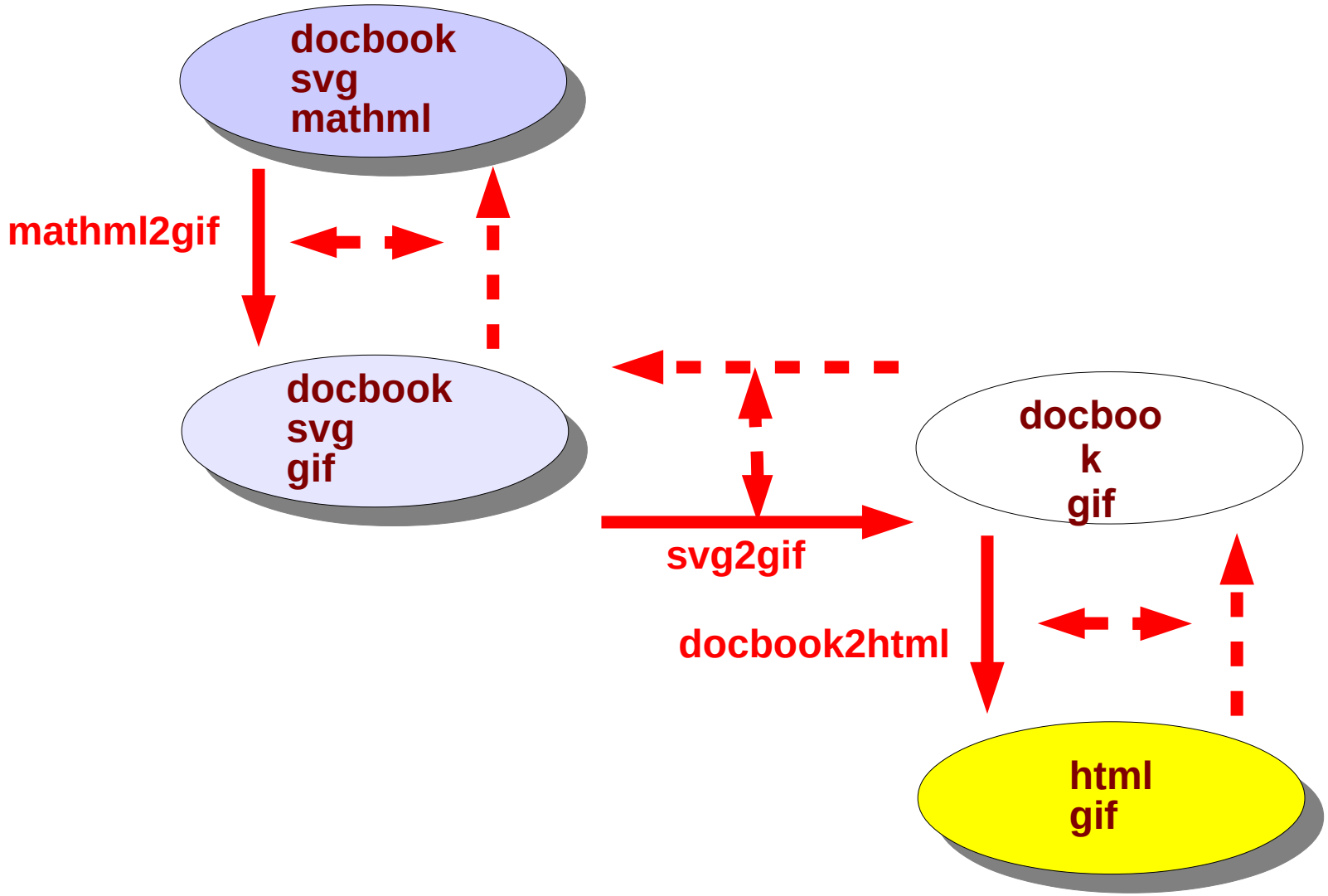


Simple ARE problems can be stacked onto each other

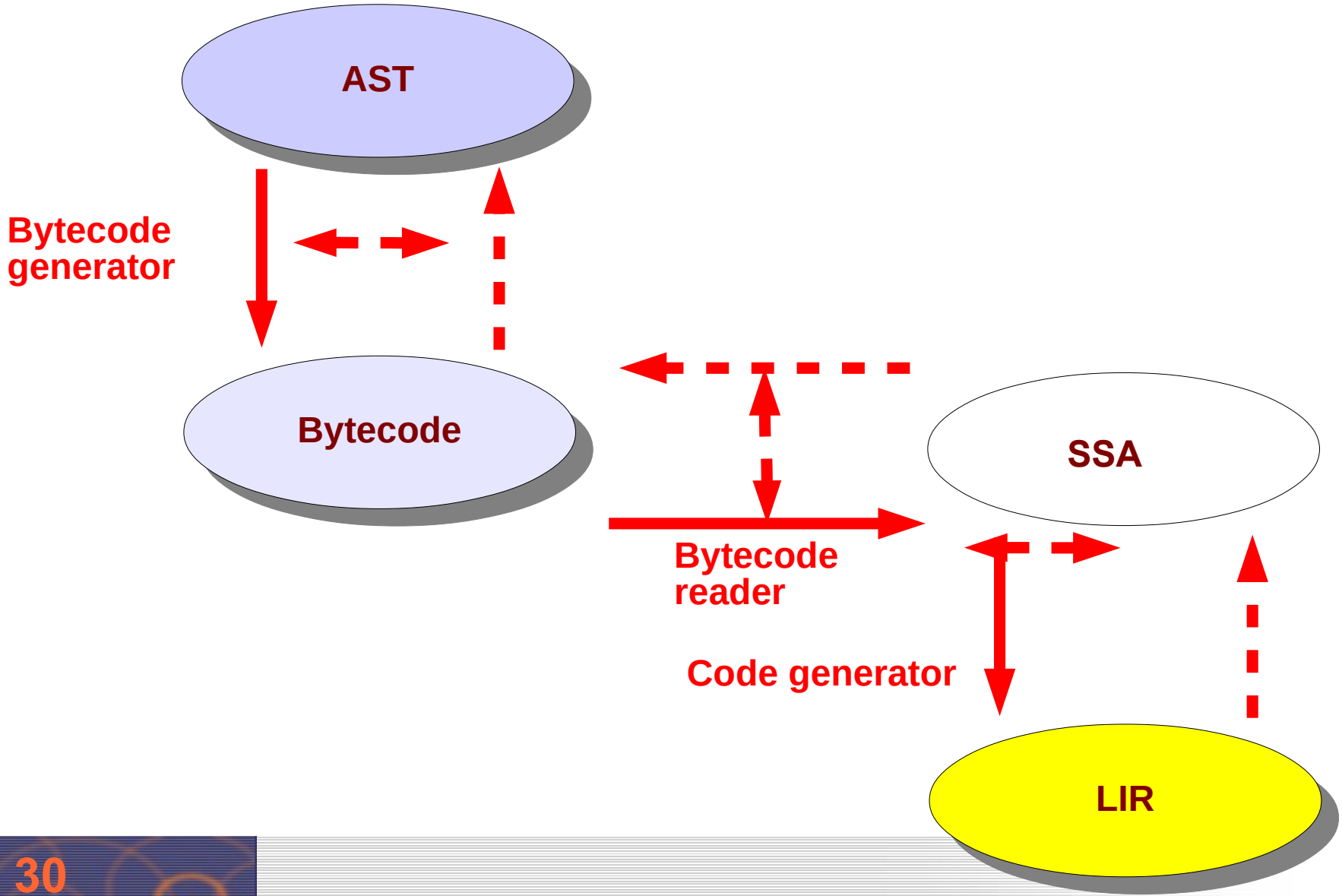
- Either in list form
- Or in tree (cactus) form

Bridge gaps between domains that are semantically very much distinct

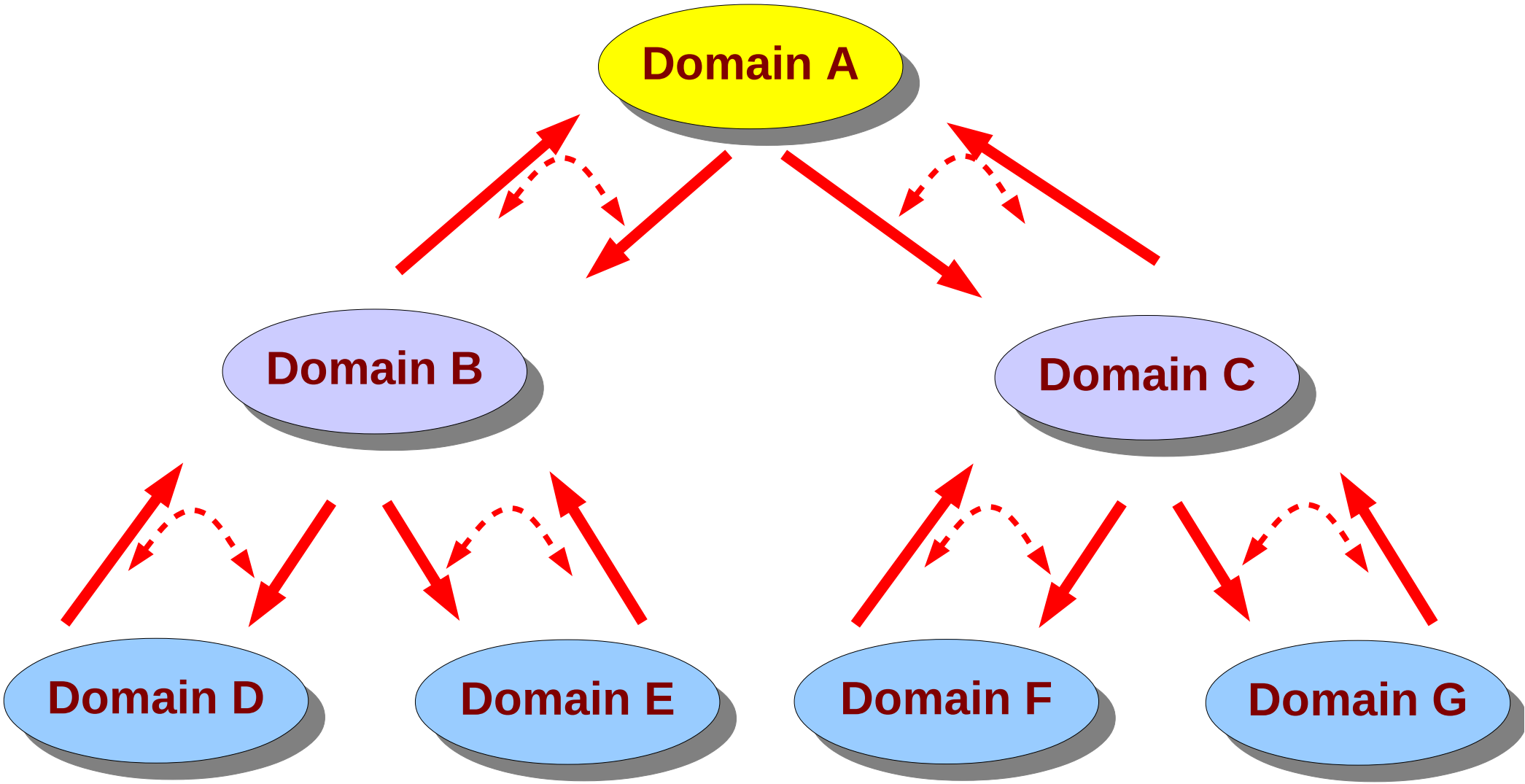
Example: XSLT Chains in Content Management Systems



Almost an Example: Compiler IRs



Cactus ARE



Extensible Languages

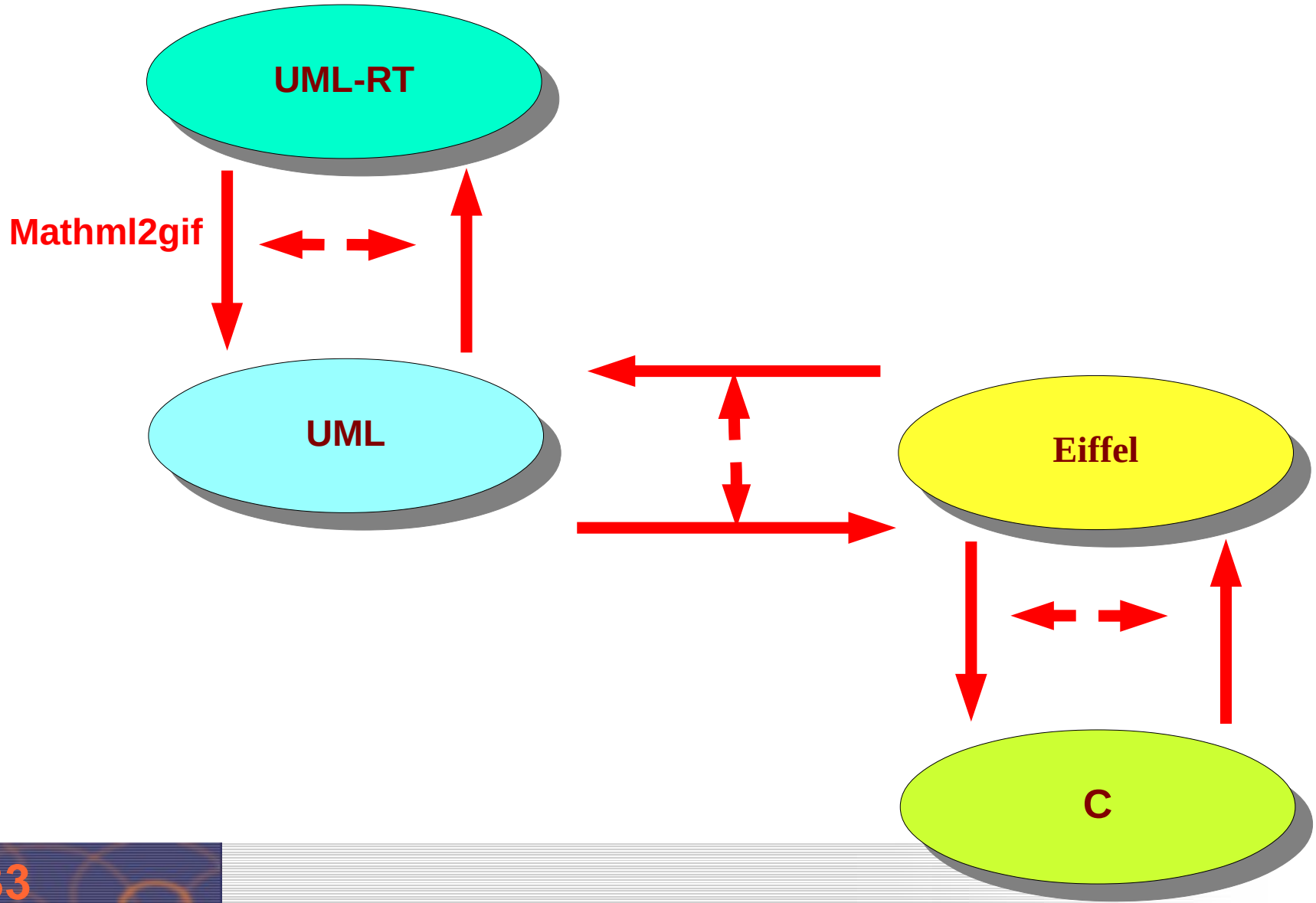
Should be ARE, but are not!

In which form should an extensible language program be maintained?

- In super language: unreadable
- In base language: better readable?? but not as concise

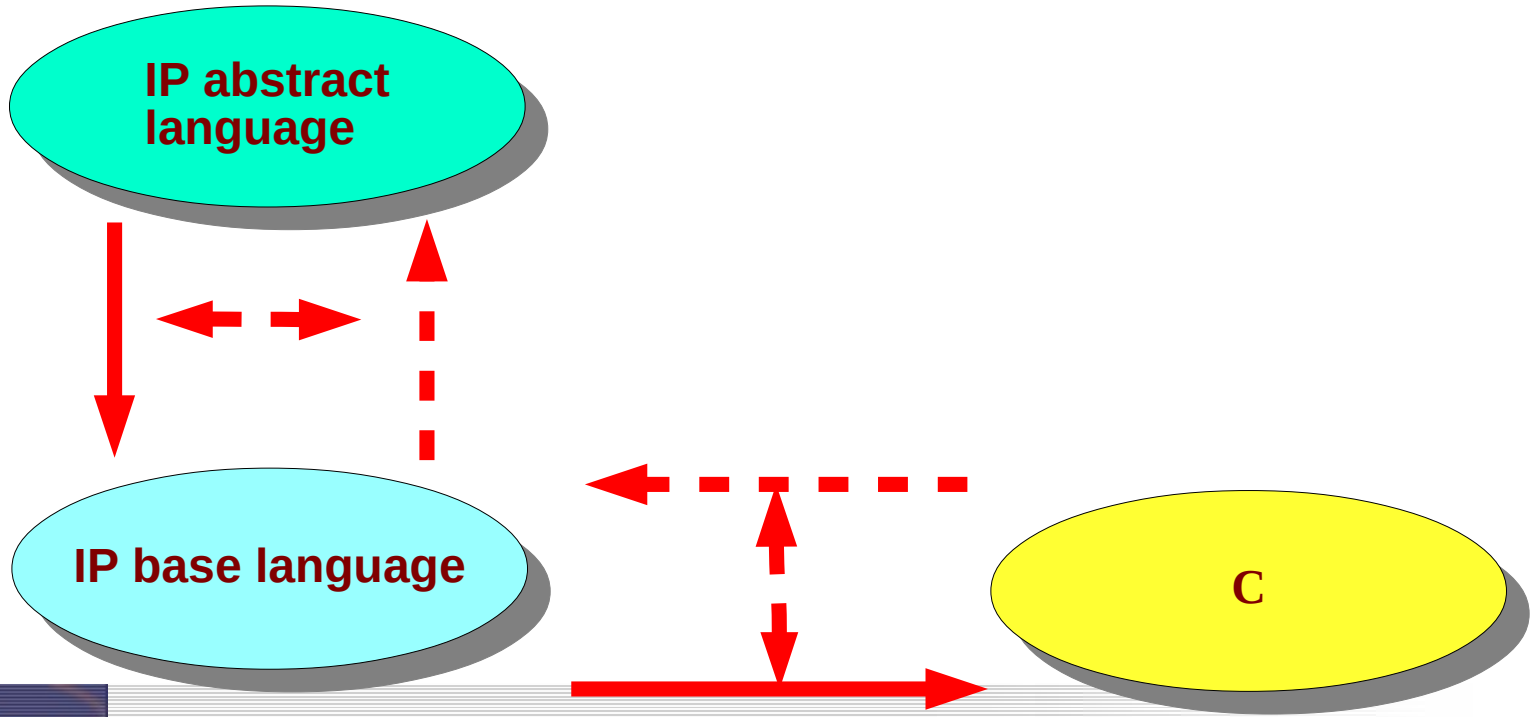
ARE allows to switch, also when the people have left the company

Domain Specific Languages with Extensible Languages

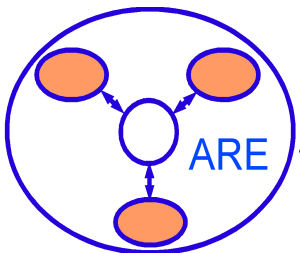


Intensional Programming of Microsoft

Is a form of extensible language for DSL
Should be inveritble

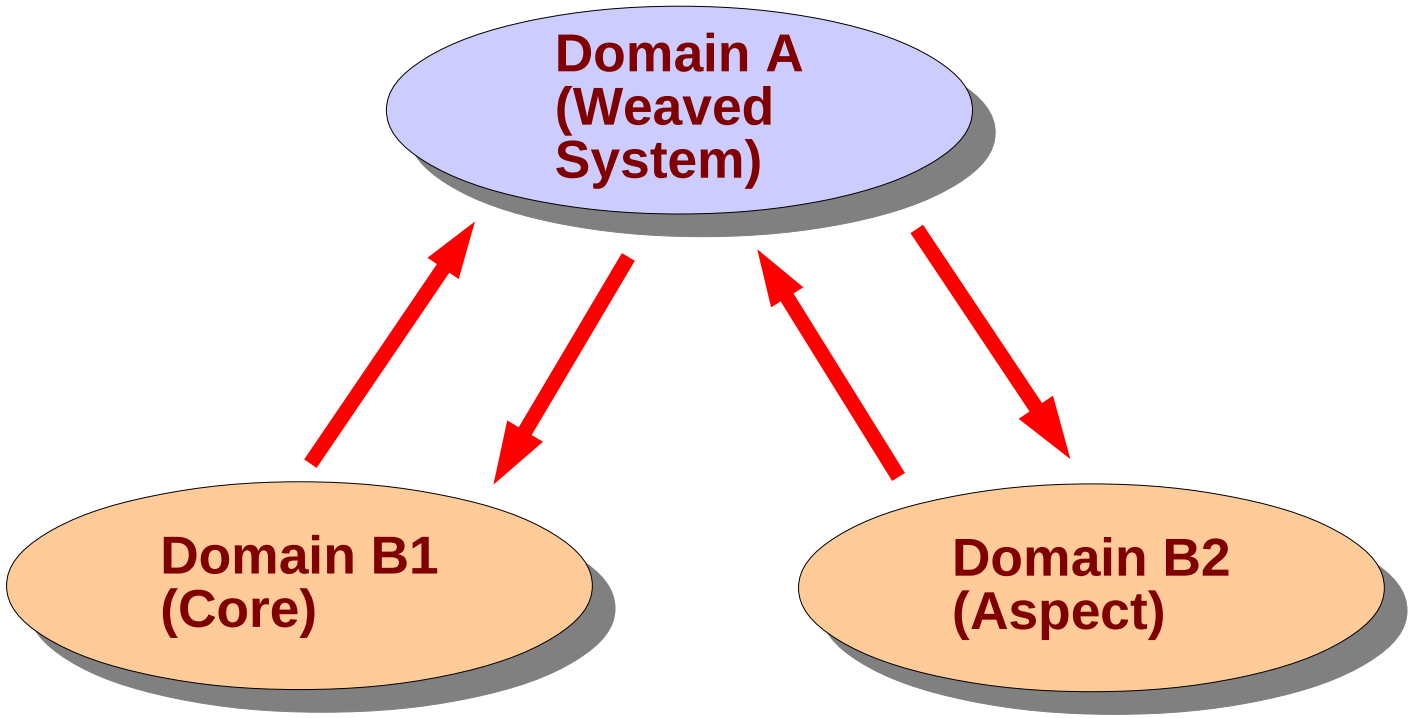


Bidirectional AOP



AOP versus MVARE

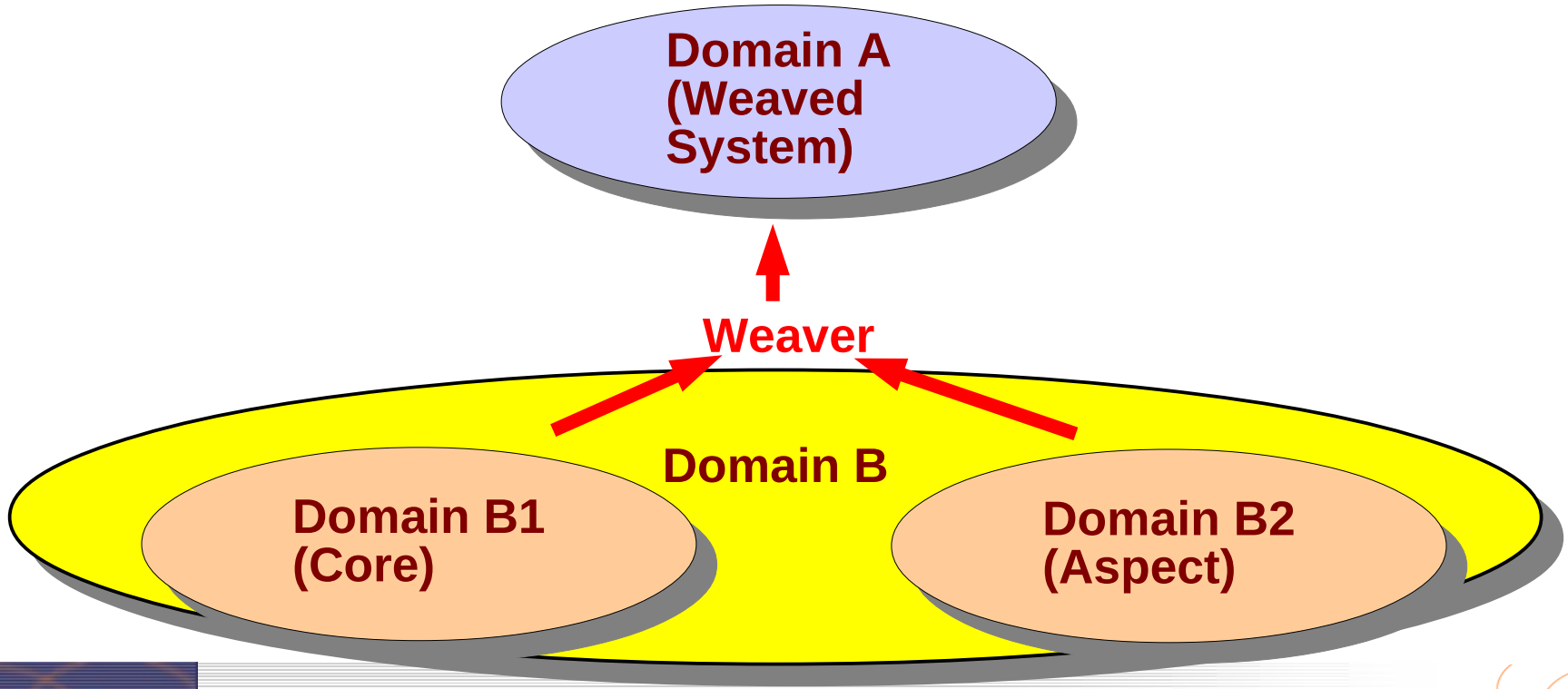
MVARE provides independent integrations and projections
In AOP, integrations depend on each other



AOP is different

Views are independent of each other
Aspects depend on the core
Aspects lack *independence*

The integrators are not independent of each other



Beaving Systems (Bidirectional AOP)

A bidirectional aspect systems are special cases of MVARE

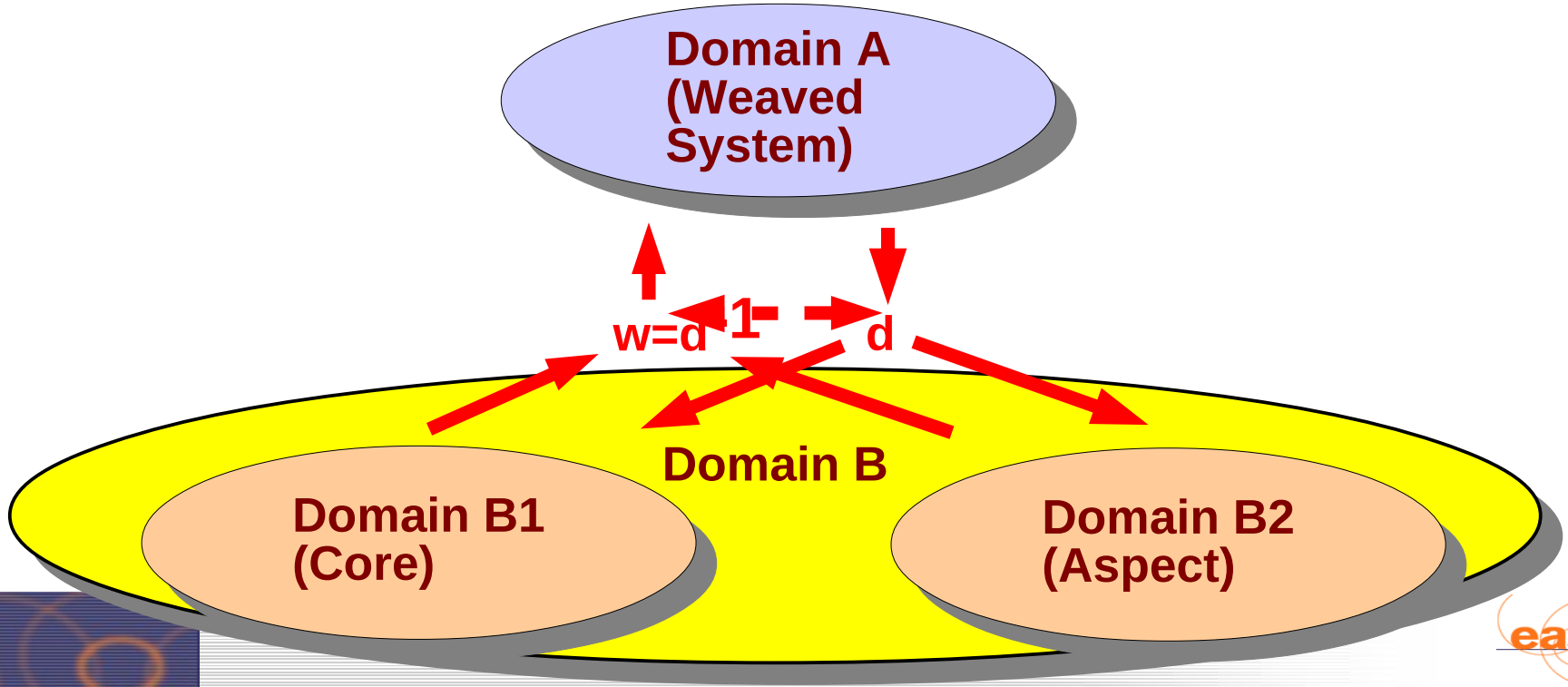
- The weaver is an integrator
- A deweaver is a projector
- Usually, deweavers are missing from AOP, so it's not a roundtrip engineering method

Such a system is called a **bidirectional weaver (beaver)**

Bidirectional Weaving (Beaving)

Bidirectional AOP is an ARE but not an MVARE

- the projections and integrations are coupled
- MVARE requires independent integrators (and independent aspects)



Advantages of Beaving

- Debugging easy compared to forward AOP
  - Tracability
  - View can be switched as desired
- Maintenance easy
  - System can be maintained in split or integrated form
- Understanding better
  - View can be switched as desired

ARE as Instance of Divide and Conquer

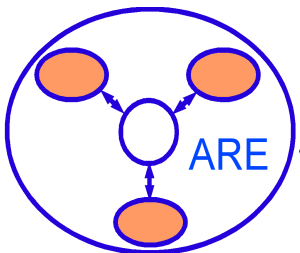
Simple ARE does not divide, but conquers by domain transformation

MVARE is Divide and Conquer for dimensional decomposition (separation of concerns)

Beaving is full AOP

- Debugging, maintenance, understanding

Incremental Processing



Incremental Evaluation with Spatially Isomorpic ARE

The transformations take time

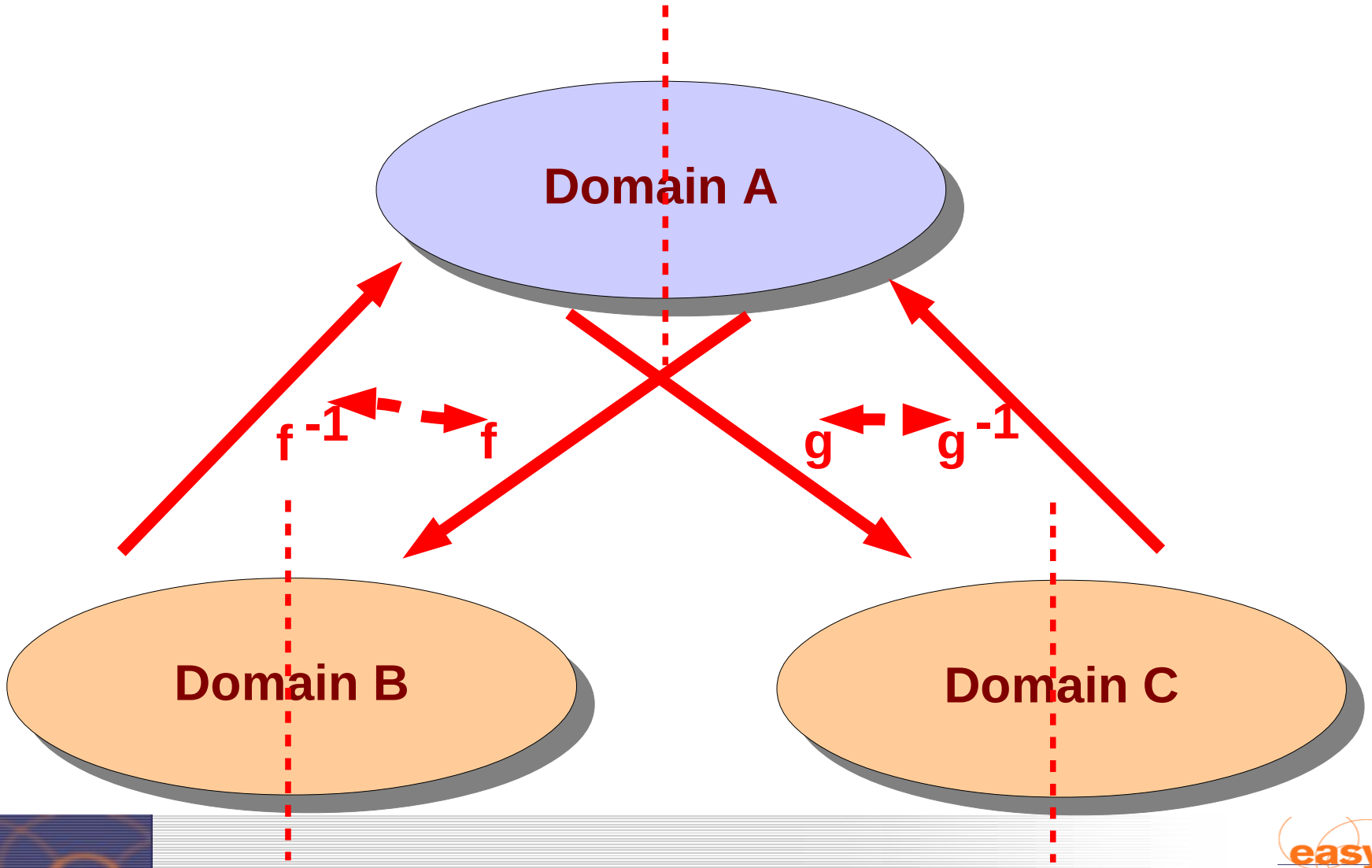
May be they can be done incrementally?

- in particular in interactive applications

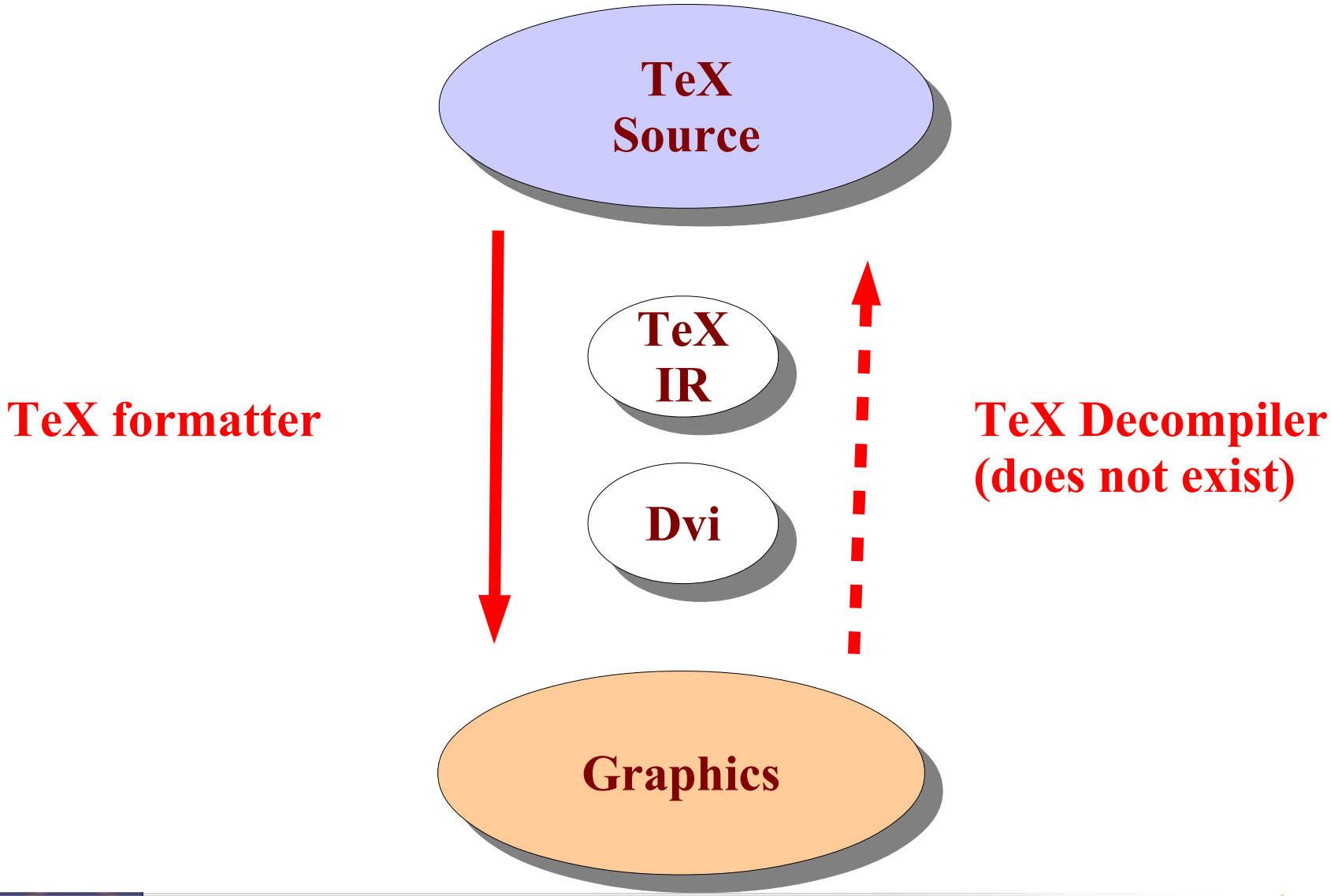
If domains can be partitioned into subdomains

- isomorphic to each other

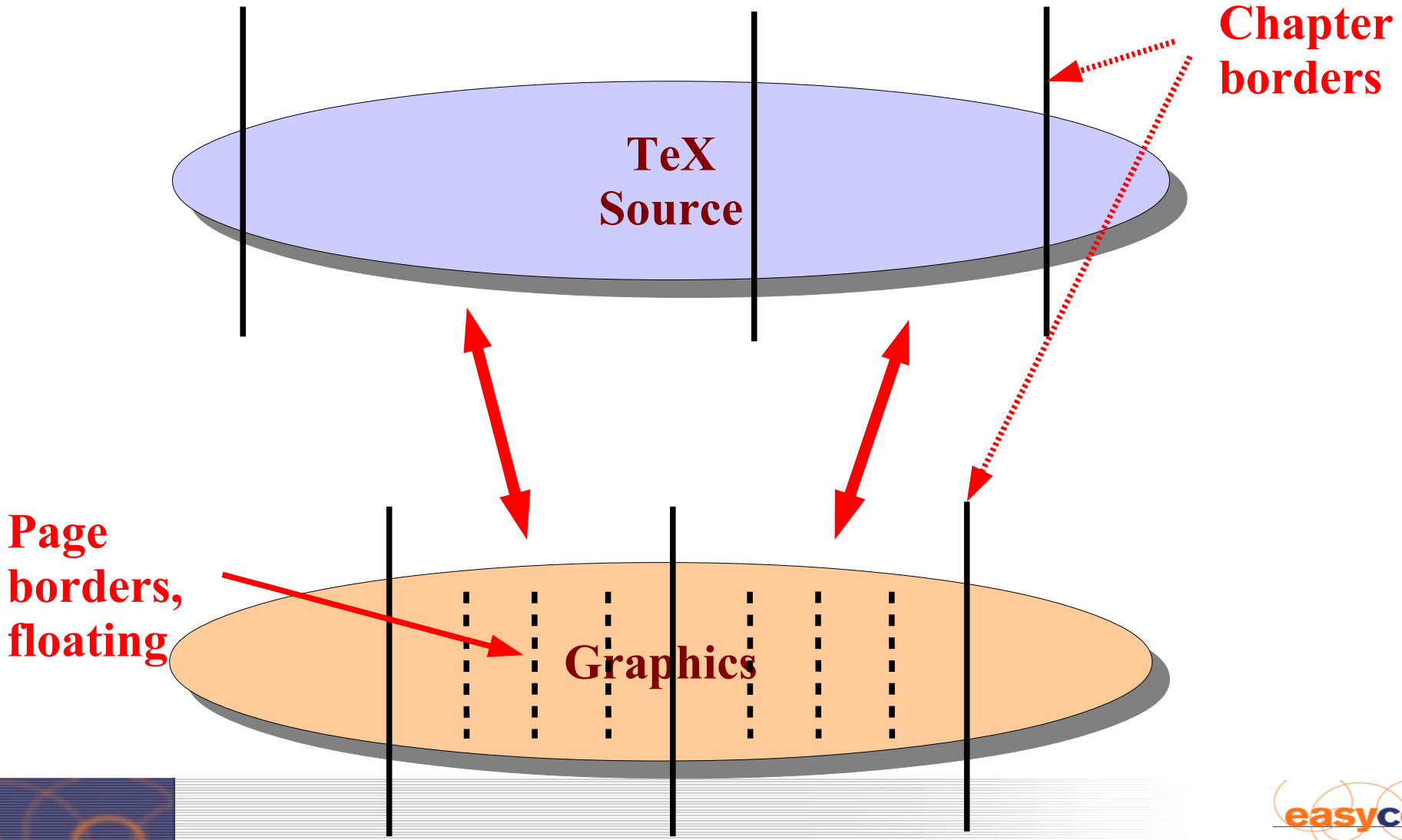
Spatial Isomorphic ARE



TeX Should Be ARE



TeX as Almost Spatial Isomorphic ARE



TeX Should Be ARE!

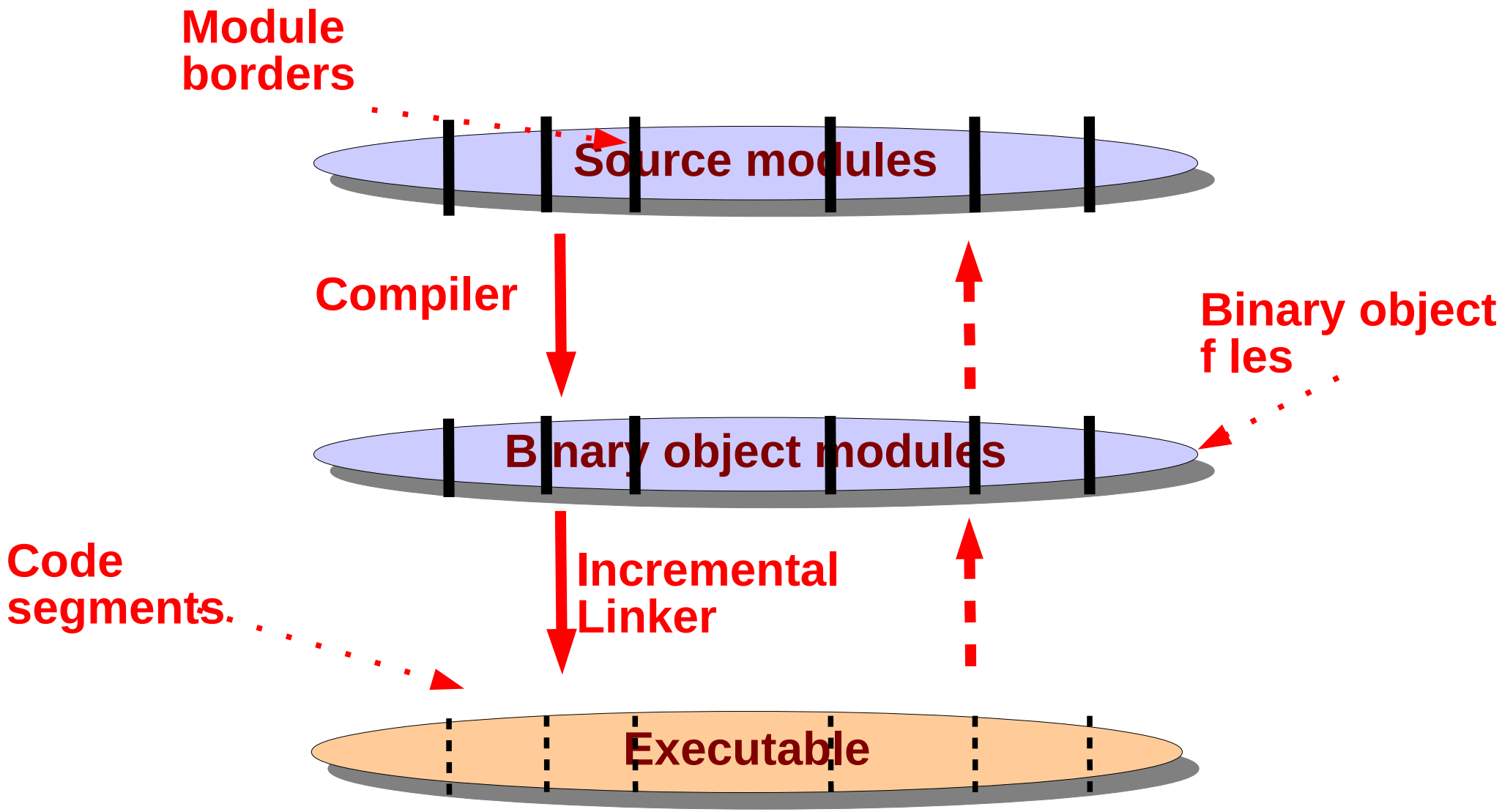
TeX formatter is not quite an ARE, but should be:

- **Incrementality**:
  - An incremental TeX needs only to reformat the current chapter
  - Chapters provide an almost isomorphic partitioning of the model and the view
- **Edit** the bitmap:
  - Deformatter should exist
  - (TeX is too powerful due to its macro system)

If TeX was an ARE system, bitmaps could be edited and still be translated into the TeX source (TeX-Emacs unification)

- and vice versa
- It would be a Chain-ARE, with several intermediate steps

Separate Compilation as Sequential SPARE



Separate Compilation as a Chain-SPARE

Separate Compilation divides the program into modules (regions)

- can be compiled separately to an object module
- Smart Recompilation on procedure level uses even finer grained regions, procedures

Also the linker can be incremental

- and hook in a newly compiled binary module into an executable

Standard, non-incremental linkers do not exploit the SPARE feature

SPARE vs MVARE

In a SPARE, usually all transformations are identical
If we use a different transformation for every partition, we get a MVARE

AC-decomposable SPARE

An AC-decomposable SPARE is a SPARE that works with a partially ordered partition of the source domain

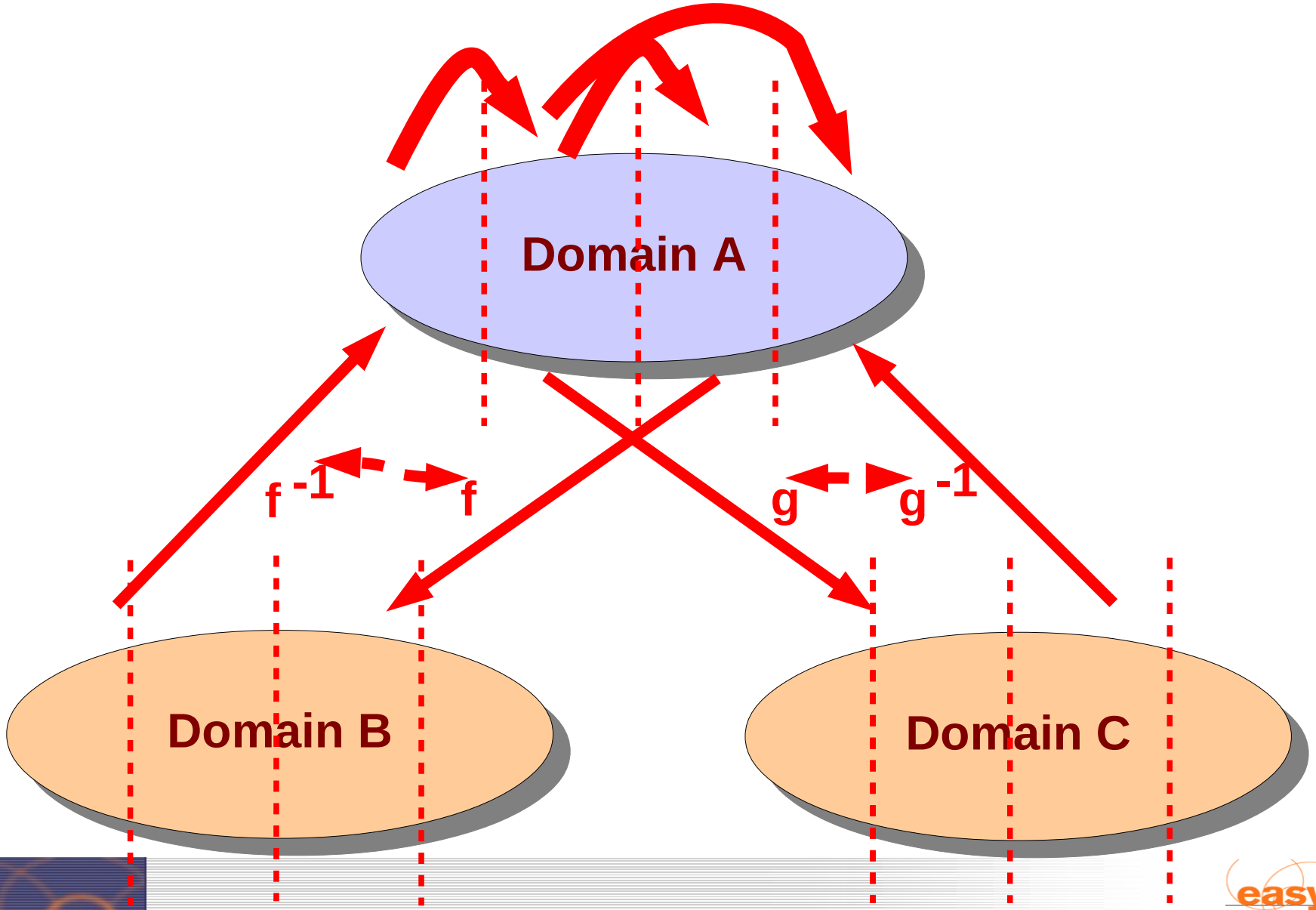
- Partitions are not independent

If the transformations are different, it is called an AC-decomposable MVARE

Partitions depend on previous partitions in the partial order

Whenever a transformation has to be performed, all previous partitions have to be transformed

AC-.SPARE



AC-SPARE

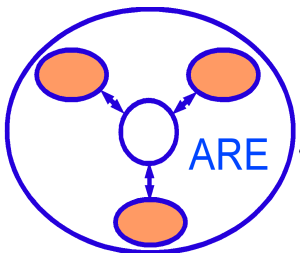
Still decomposable, i.e., incremental
But only incremental in the depth of the partially ordered relation!
OAGs are powerful special forms of AC relations

- Ordering means that the partial dependencies of the attribute relations can be ordered statically
- AC-decompability can mean that OAG algorithms are employed

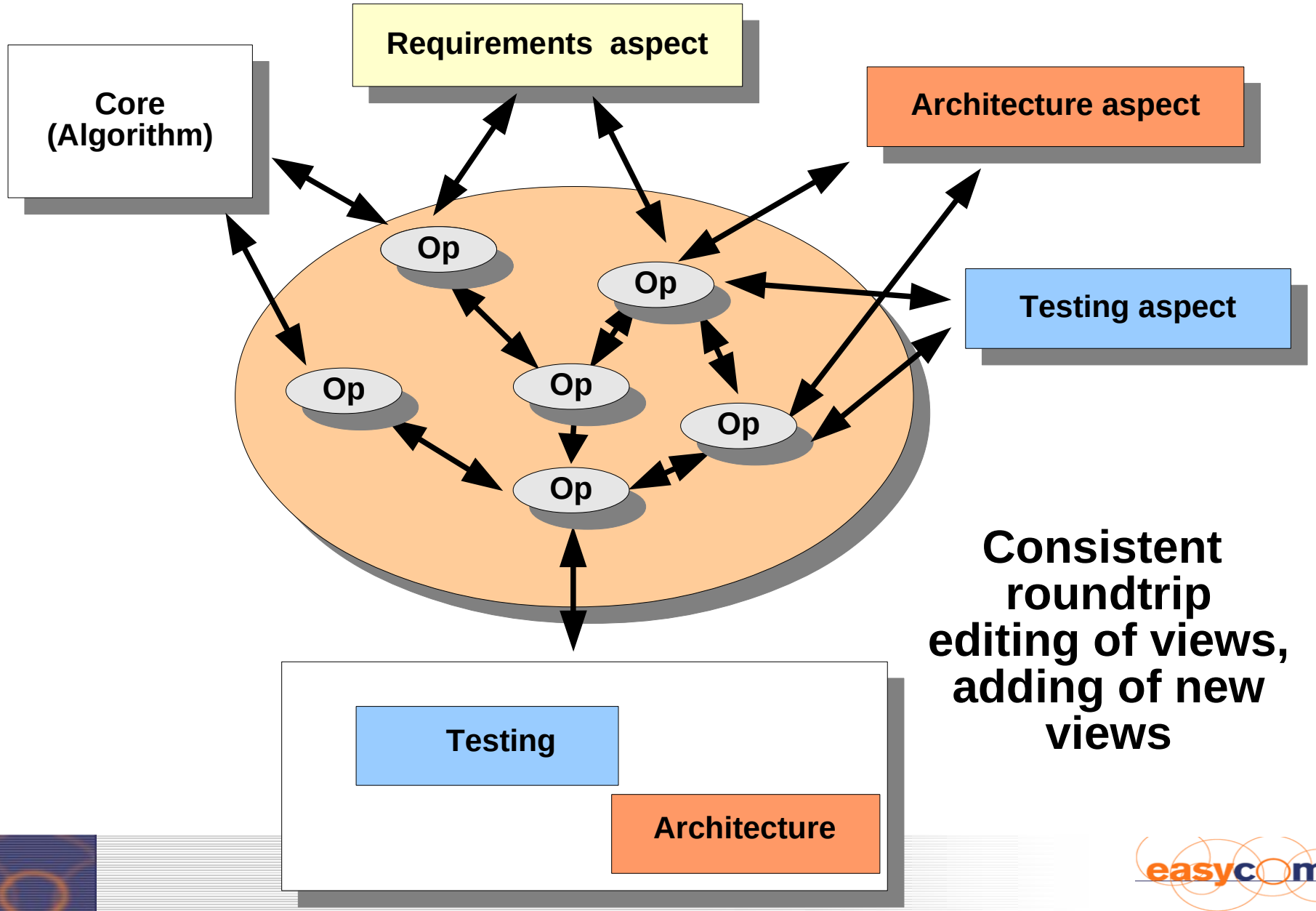
OAG-HAGs are AC-SPAREs

Integrational Software Engineering

And why you will want to use it..



Automatic Roundtrip Engineering in an Environment for Integrational SE

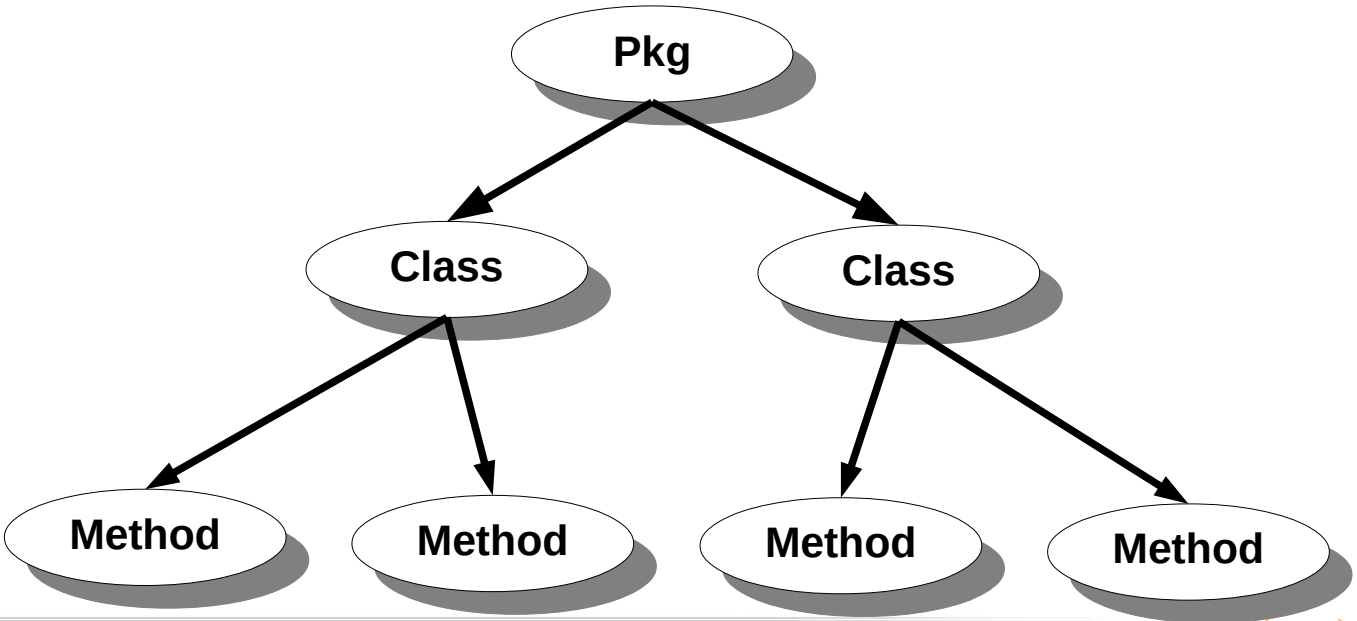


Consistent roundtrip editing of views, adding of new views

Example: Scope Trees

Scope trees are structured containers for code and its attributes

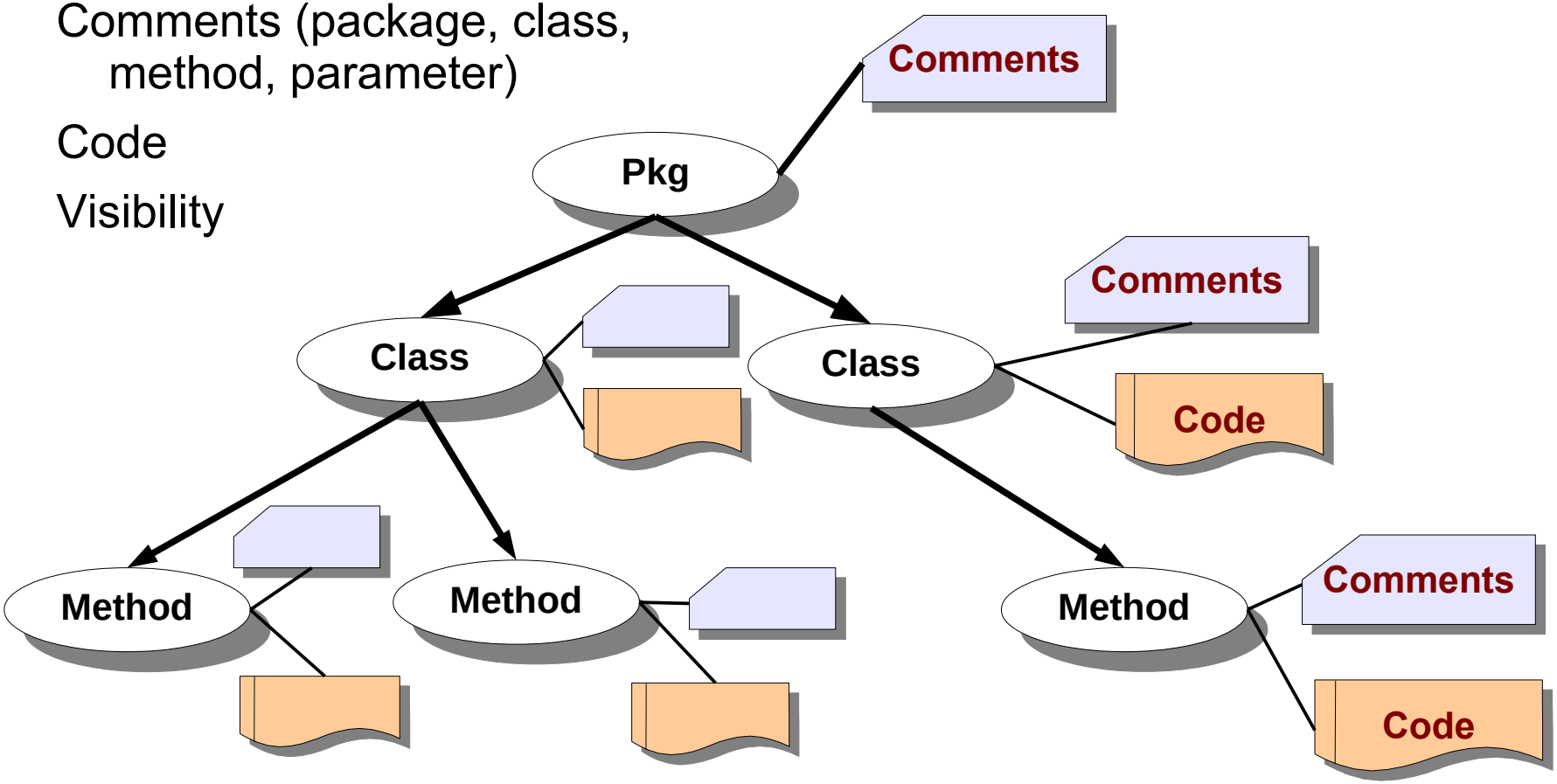
- Package tree of a program (static structuring)
- Scope tree of a program (static structuring)



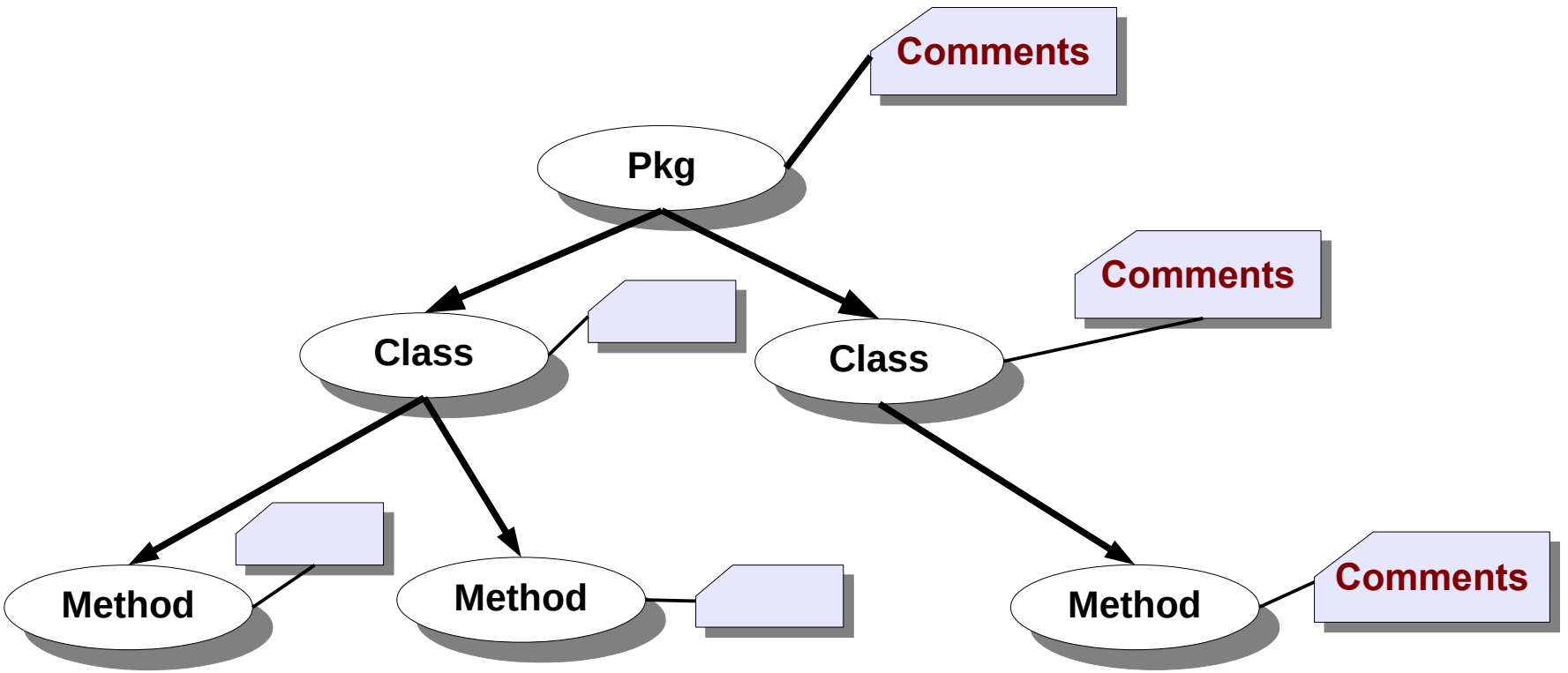
Scope Trees with Attributes

Attributes

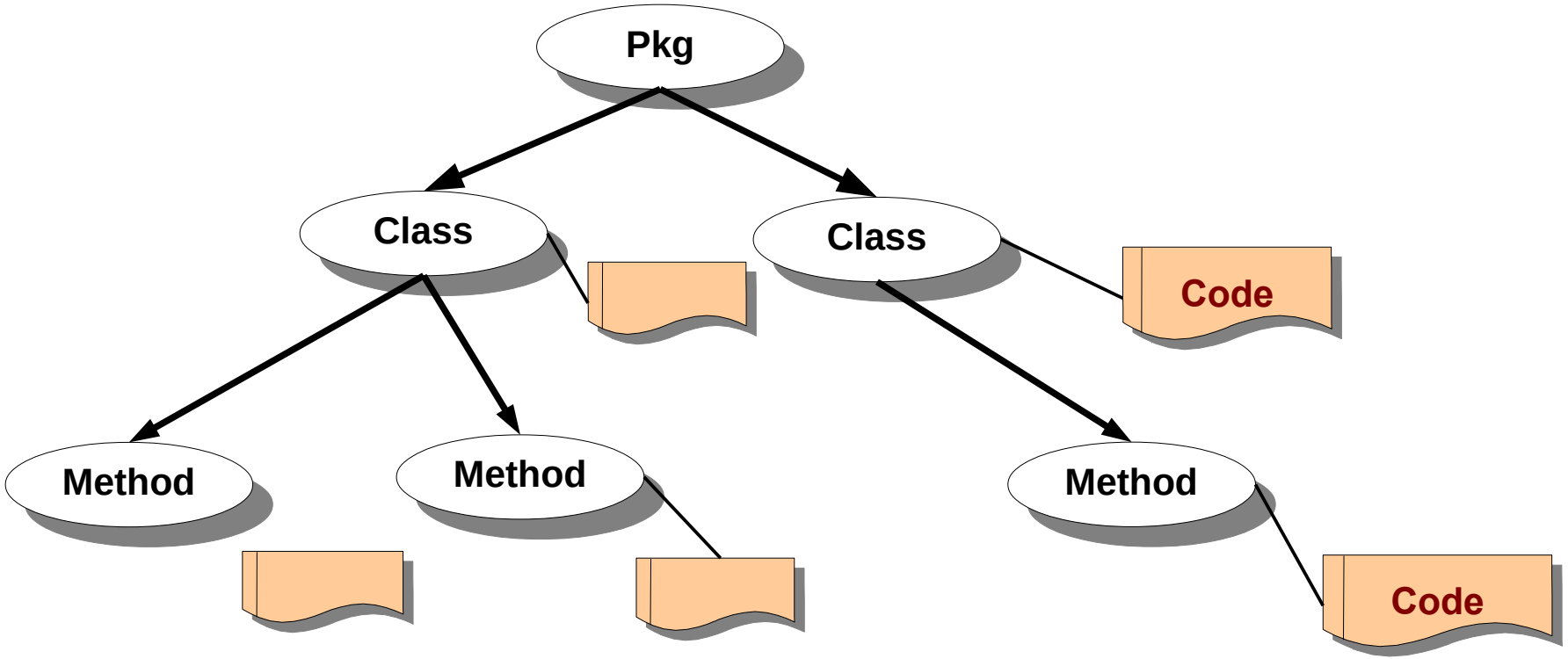
- Comments (package, class, method, parameter)
- Code
- Visibility



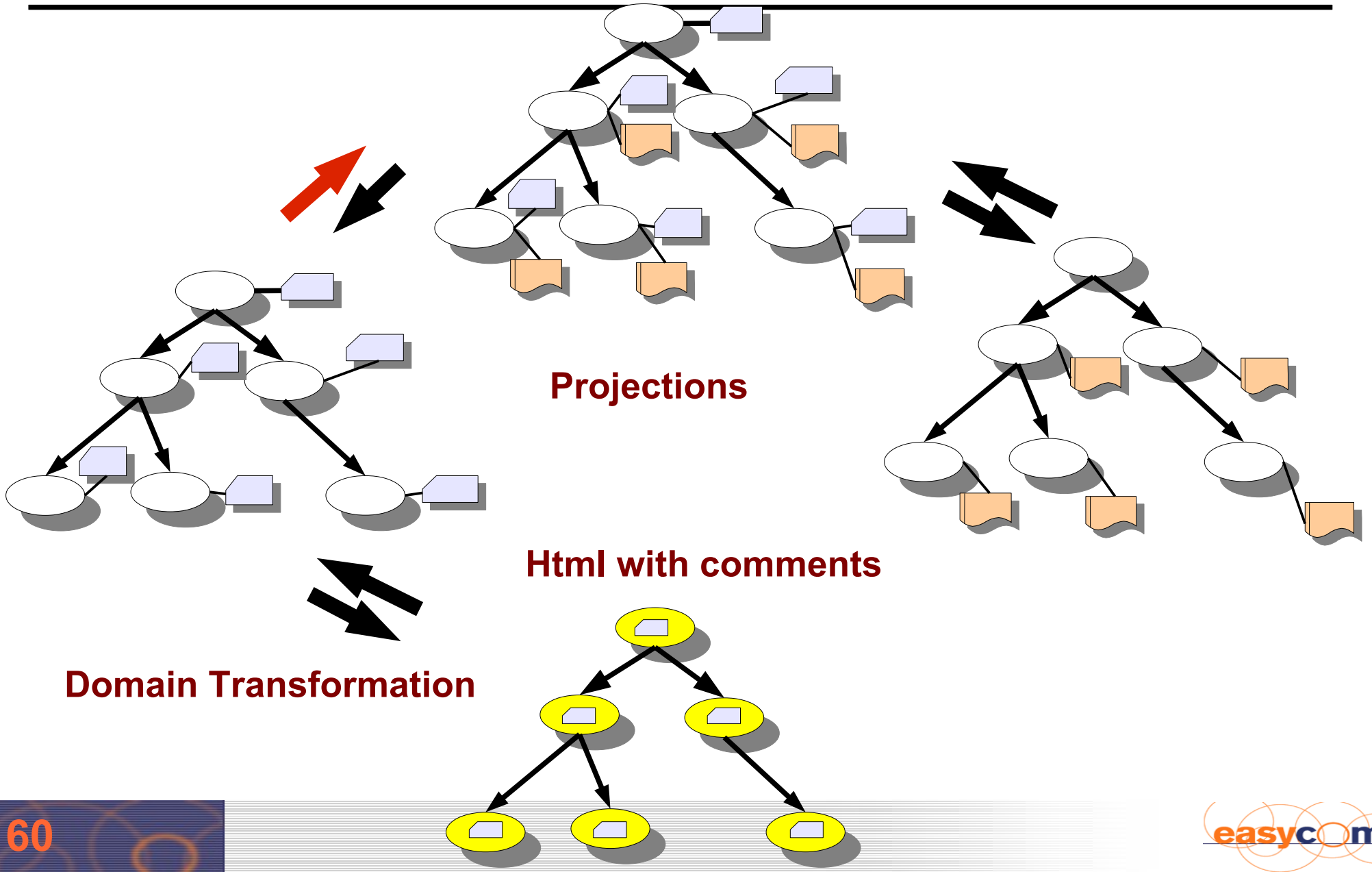
Projecting A Scope Tree For Comment Attribute



Projecting A Scope Tree For Code Attribute



JavaDoc Can Be An ARE Application



Example JavaDoc

- JavaDoc is a projection from a system of structural tree, code and commments
  - To tree with comments
- JavaDoc can be modeled with a generated inverse DPO graph rewrite system
  - Names of packages, classes, methods are used as identification tags
  - The system is invertible

Realization With Extended DPO

DPO (double pushout) graph rewriting has inverses
DPO has restrictions when nodes are deleted

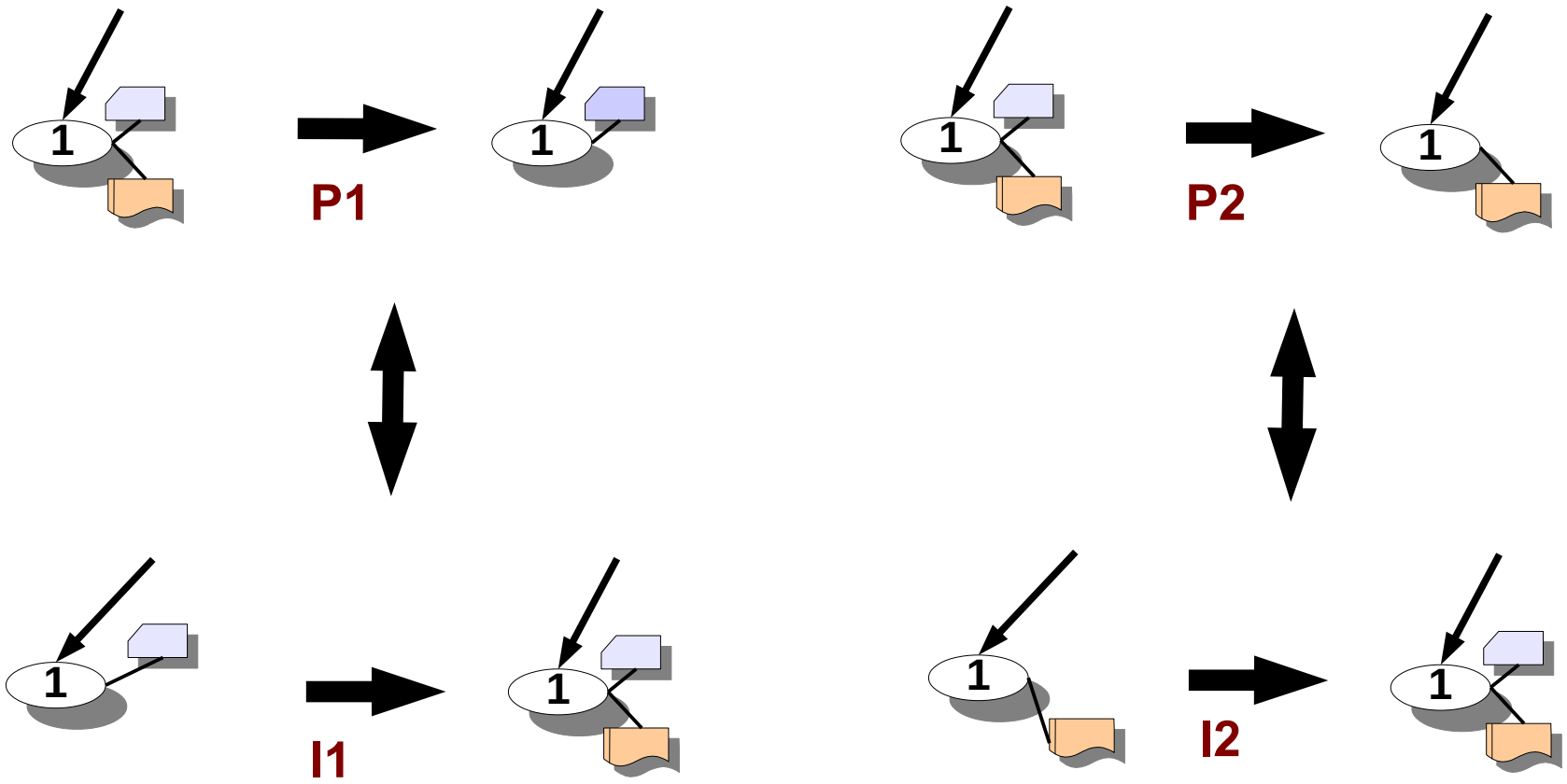
- Cannot correct “hidden” edges

Extended DPO with memoization of redexes

- Introduction of “ghost” objects
- Deleted nodes are memorized, until integration regenerates model

[Assmann/Ludwig GCSE 99]
[Larsson/Burbeck 02 master's thesis]
[Larsson/Burbeck 03 MDAFA]

A DPO System For Views on Comments and Code



Realization ctd.

ALL methods can be used that produce invertible transformations
Invertible TRS
Invertible Forward AGs
More....

Benefits of the ARE Architectural Style

Simple ARE does not divide, but wins by domain transformation
MVARE is Divide and Conquer for dimensional decomposition (separation of concerns)

- The inverse generators make it simpler

Chain-ARE is for minimizing the semantic gaps
SPARE additionally divide into spatial regions, to make it incremental (spatial refinement)
Bi-AOP is *better* AOP

- Debugging, maintenance, understanding
- MVARE is almost AOP, but simpler

Why You Will Earn A Lot of Money

ARE styles are based on

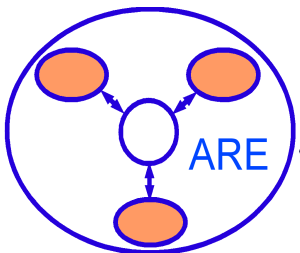
- Compiler technology
- Rewrite system technology
- Formal transformation approaches

Domain transformations and D&C are THE major problem solving methods of mankind

Compilers are the only available tools for domain transformations and D&C

Hence.....

Compilers Will Live Forever



RISE

The End

Software Composition (SC 2003)
MDAFA 2003 Twente University
MDAFA 2004 Linköping University
uweas@ida.liu.se
www.easycomp.org EASYCOMP project on software composition