ADR at Work, part I (Past)

Alberto Lluch Lafuente[1] (speaker)
Roberto Bruni[1], Ugo Montanari[1], Emilio Tuosto[2] (contributors)

[1]PISA, [2]ULEICES
{bruni,lafuente,ugo}@di.unipi.it, et52@mcs.le.ac.uk

Sensoria Workshop München, March 11-14, 2008

What isn't ADR

WIKIPEDIA
The Free Encyclopedia

navigation
- Main Page
- Contents
- Featured content
- Current events
- Random article

interaction
- About Wikipedia
- Community portal
- Recent changes
- Contact Wikipedia
- Donate to Wikipedia
- Help

search

Go Search

toolbox
- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link
- Cite this page

ADR is a three-letter acronym that may refer to:

- Académie de Roberval, a school in Montreal, Canada
- short for ***A****ccord européen relatif au transport international des marchandises* ***d****angereuses par* ***r****oute*, also known as the European Agreement concerning the International Carriage of Dangerous Goods by Road
- Adiabatic Demagnetisation Refrigeration
- Adria Airways, an airline of Slovenia (ICAO code: ADR)
- Advanced Digital Radio Testing Service
- Advanced Dungeons & Rabbits, a Role Playing Game for phpBB
- Adverse drug reaction
- Airdrie railway station, United Kingdom (National Rail code: ADR)
- Alter Der Ruine, a power noise group from Tucson, Arizona
- Alternative Democratic Reform Party, a political party in Luxembourg
- Alternative dispute resolution
- American Depositary Receipt, a method of trading foreign stocks
- Andrews Municipal Airport, located in South Carolina (IATA code: ADR)
- Applied Data Research
- Artificial Disc Replacement
- Astra Digital Radio
- Australian Design Rules, a set of construction standards for road registered vehicles in Australia
- Automated Dialogue Replacement or Additional Dialogue Recording, also known as "dubbing"
- Average daily rate, a common lodging industry statistic
- Azerbaijan Democratic Republic

adr may also mean:

- The **adr** microformat, a sub-set of the hCard microformat.

Problem statement

Main problems ADR faces:

- P1 := Build architectures with structural properties ϕ.
  - P1.SOC := applications have holes (services).
- P2 := Reconfigure architectures preserving ϕ.
  - P2.SOC := Holes (services) to be reconfigured internally.

Problem statement

Main problems ADR faces:

- P1 := Build architectures with structural properties ϕ.
  - P1.SOC := applications have holes (services).
- P2 := Reconfigure architectures preserving ϕ.
  - P2.SOC := Holes (services) to be reconfigured internally.

Some flaws of existing approaches

- P1 via ... Drop&Bind ingredients, check ϕ: tedious.
- P1 via ... Bounded SAT: no guidance, trial&error.
- P2 via ... Show ϕ-preservation: manual.
- P2 via ... Monitor ϕ-preservation: no guarantee.

Principles of ADR

Architectural **D**esign **R**ewriting?

- **A**lgebra of designs
  - Type T_ϕ set of architectures that satisfy $\phi \rightsquigarrow$ P1.
  - Set of design productions (operations, inductive definitions).
- **D**omain
  - Designs: graphs with interfaces.
  - Partial designs: designs with holes $\rightsquigarrow$ P1.SOC.
- **R**ewriting
  - Rewrite design terms (not designs) $d : T \longrightarrow d' : T \rightsquigarrow$ P2.
  - Based on conditional term rewriting, SOS $\rightsquigarrow$ P2.SOC.

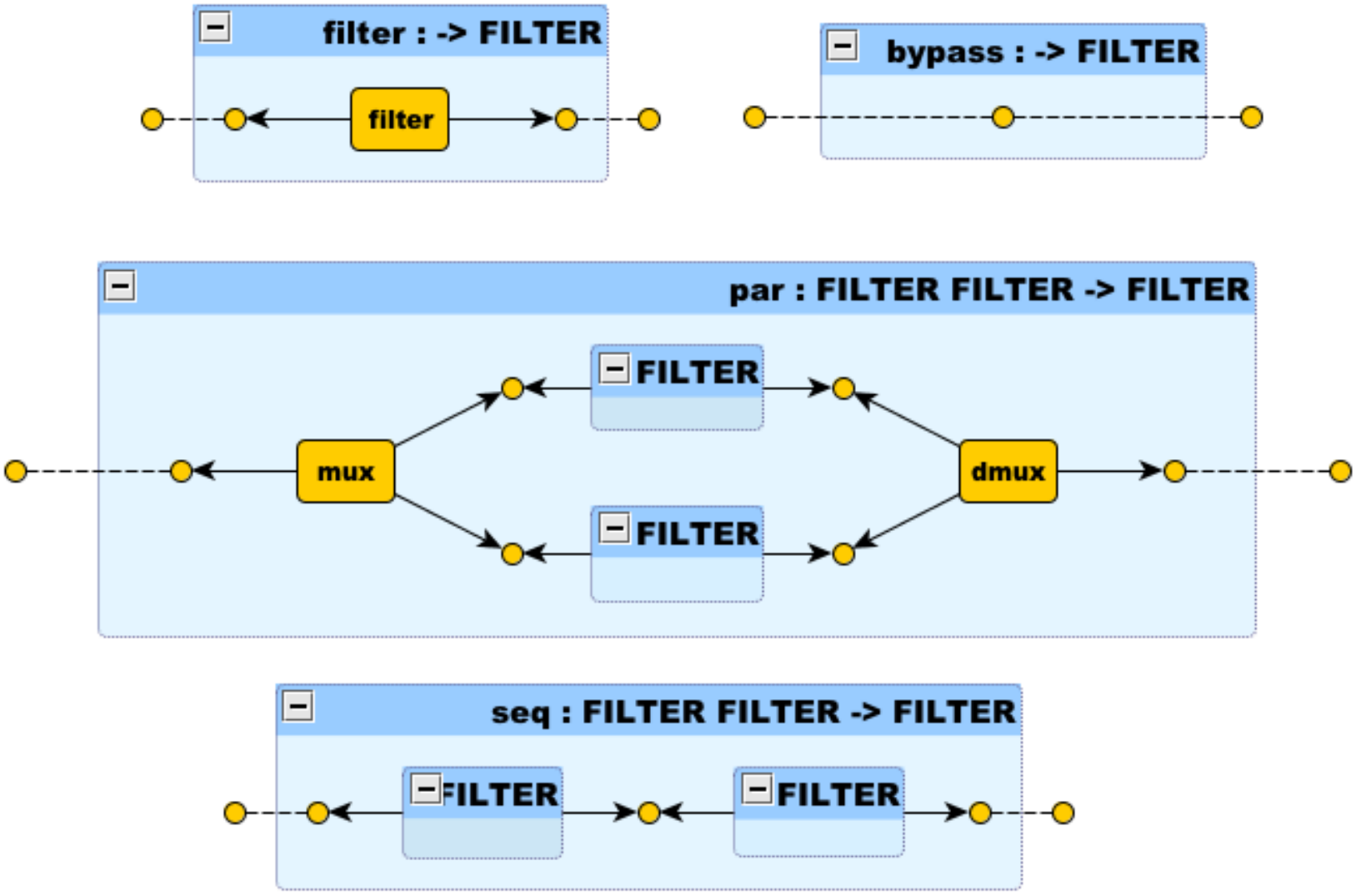
Principles of ADR

Architectural **D**esign **R**ewriting?

- **A**lgebra of designs
  - Type T_ϕ set of architectures that satisfy $\phi \rightsquigarrow$ P1.
  - Set of design productions (operations, inductive definitions).
- **D**omain
  - Designs: graphs with interfaces.
  - Partial designs: designs with holes $\rightsquigarrow$ P1.SOC.
- **R**ewriting
  - Rewrite design terms (not designs) $d : T \longrightarrow d' : T \rightsquigarrow$ P2.
  - Based on conditional term rewriting, SOS $\rightsquigarrow$ P2.SOC.

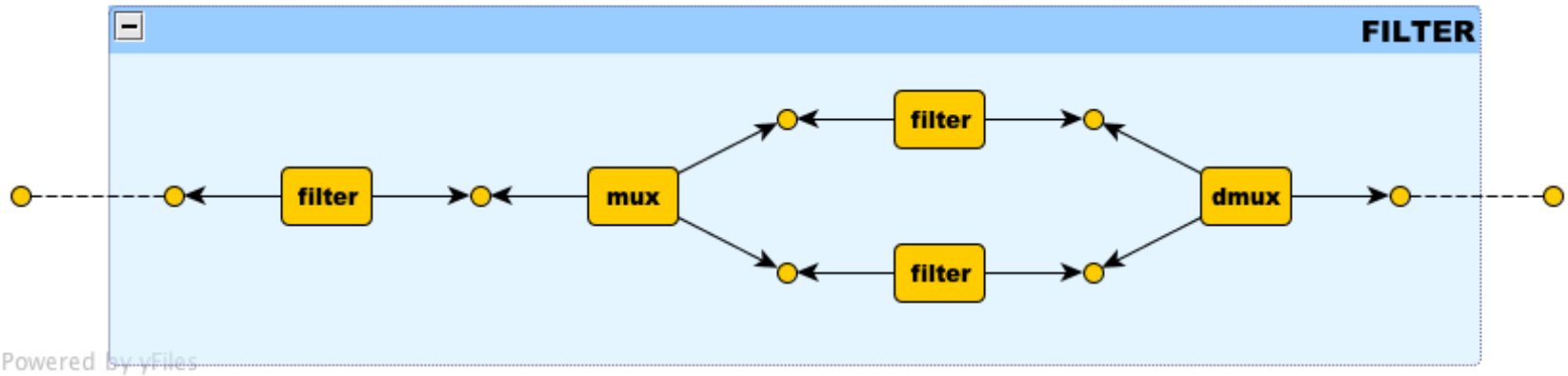
No panacea: not everything can be ADRized, but you can be happy if you manage to capture part of your problem.

Ex1: Pipes-and-Filters (A)



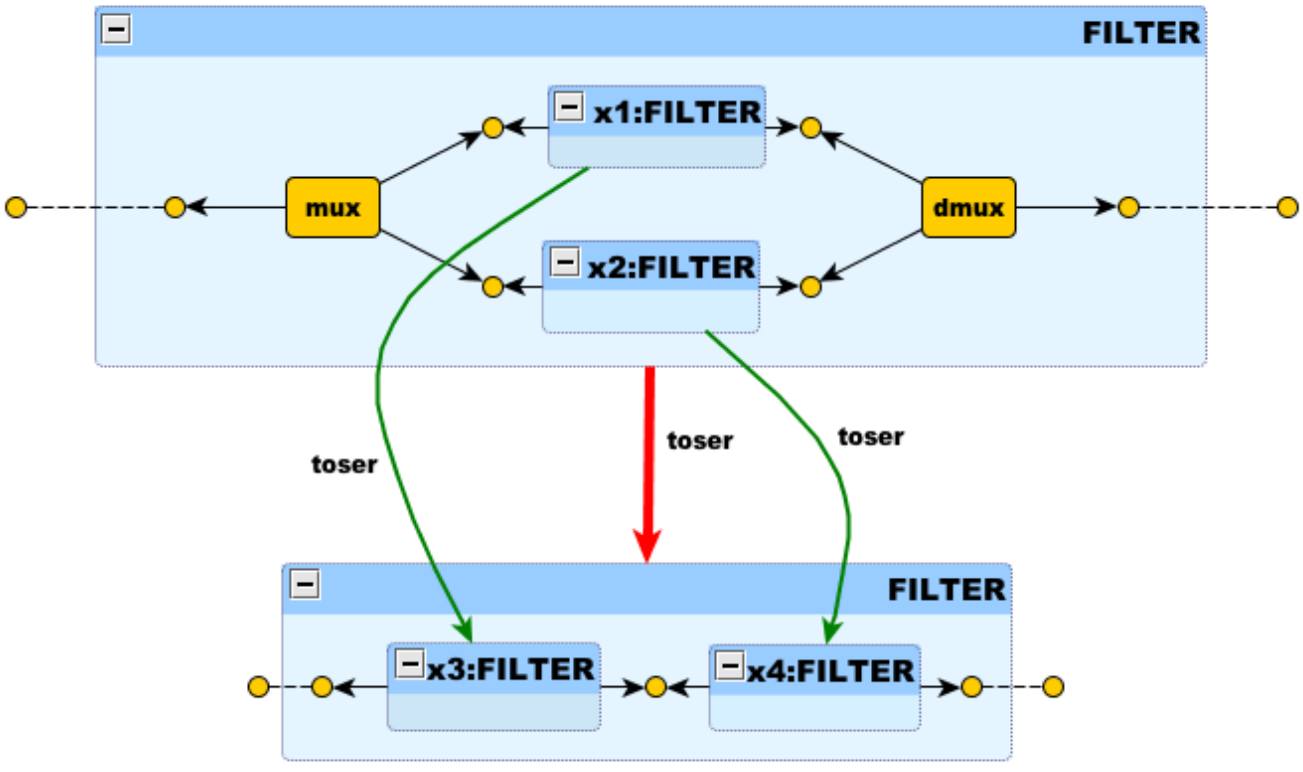
Ex1: Pipes-and-Filters (D)

```
seq(filter1,par(filter2,filter3))
```



Ex1: Pipes-and-Filters (R)

Serializing a filter (and its subfilters)



- *More in [WRLA'08]*

ADR prototype in Maude

Why Maude?

- Rewrite Theories match ADR features
  - Types as sorts (+membership).
  - Design productions as operations (+axioms).
  - Conditional term rewriting.
  - Structural operational semantics (standard encoding).
- Built-in Tools: LTL Model Checker, etc.

ADR prototype in Maude

Why Maude?

- Rewrite Theories match ADR features
  - Types as sorts (+membership).
  - Design productions as operations (+axioms).
  - Conditional term rewriting.
  - Structural operational semantics (standard encoding).
- Built-in Tools: LTL Model Checker, etc.

Playing with ADR in Maude

- Implement ADR models.
- Specify properties in some logics (e.g. à la VLRL, MSO).
- Simulate modelling activities (e.g. refinement, model finding).
- Analyse models (e.g. via model checking).
- Export graphs to dot and XML graph formats.

- *More in [WRLA'08]*

Analysis example

We require some ordering constraints ϕ among filters.

```
Maude> srew FClient-nt using modelCheck(phi)
Solution 7
result FClient: wrap(par(filter(1), Mux-nt, Dmux-nt ...
```

Analysis example

We require some ordering constraints ϕ among filters.

```
Maude> srew FClient-nt using modelCheck(phi)
Solution 7
result FClient: wrap(par(filter(1), Mux-nt, Dmux-nt ...
```

Does the 7th solution preserve some other ordering constraints ψ .

```
Maude> red modelCheck(sol7,[]psi) .
result ModelCheckResult:
counterexample...
```

Analysis example

We require some ordering constraints ϕ among filters.

```
Maude> srew FClient-nt using modelCheck(phi)
Solution 7
result FClient: wrap(par(filter(1), Mux-nt, Dmux-nt ...
```

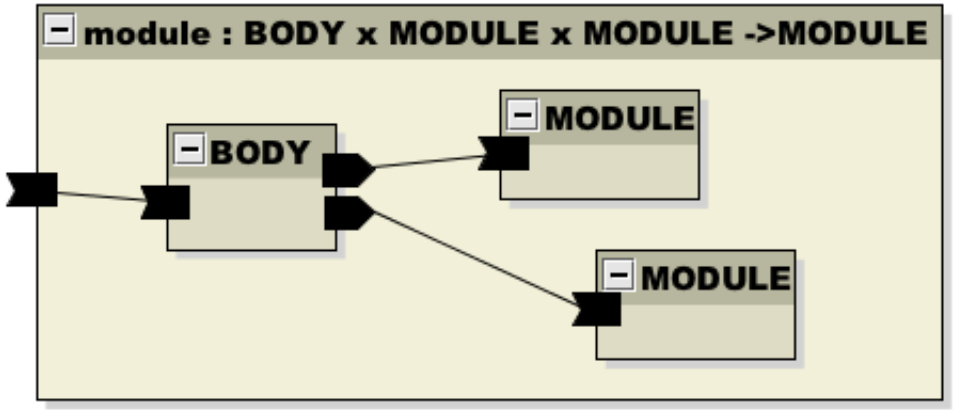
Does the 7th solution preserve some other ordering constraints ψ .

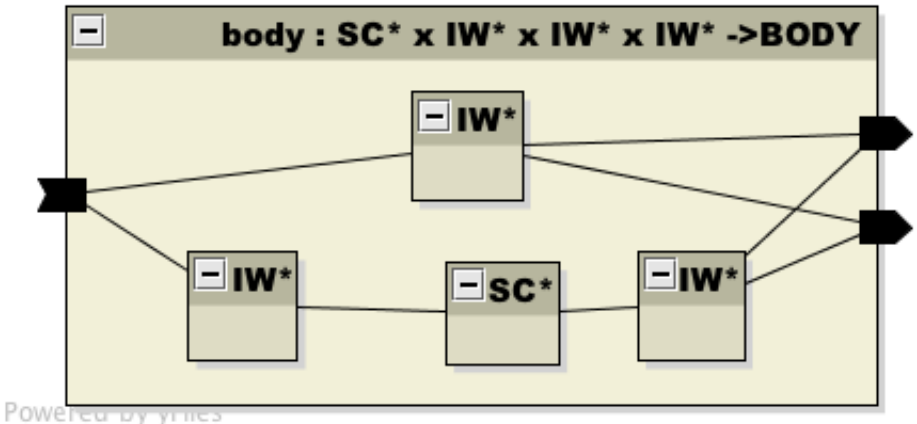
```
Maude> red modelCheck(sol7,[]psi) .
result ModelCheckResult:
counterexample...
```

We ask for an architecture satisfying ϕ and preserving ψ.

```
Maude> srew FClient-nt using modelCheck(phi /\ []psi)
Solution 3
result FClient: wrap(seq(filter(0), par(filter(1), ...
```

Ex2: SRML



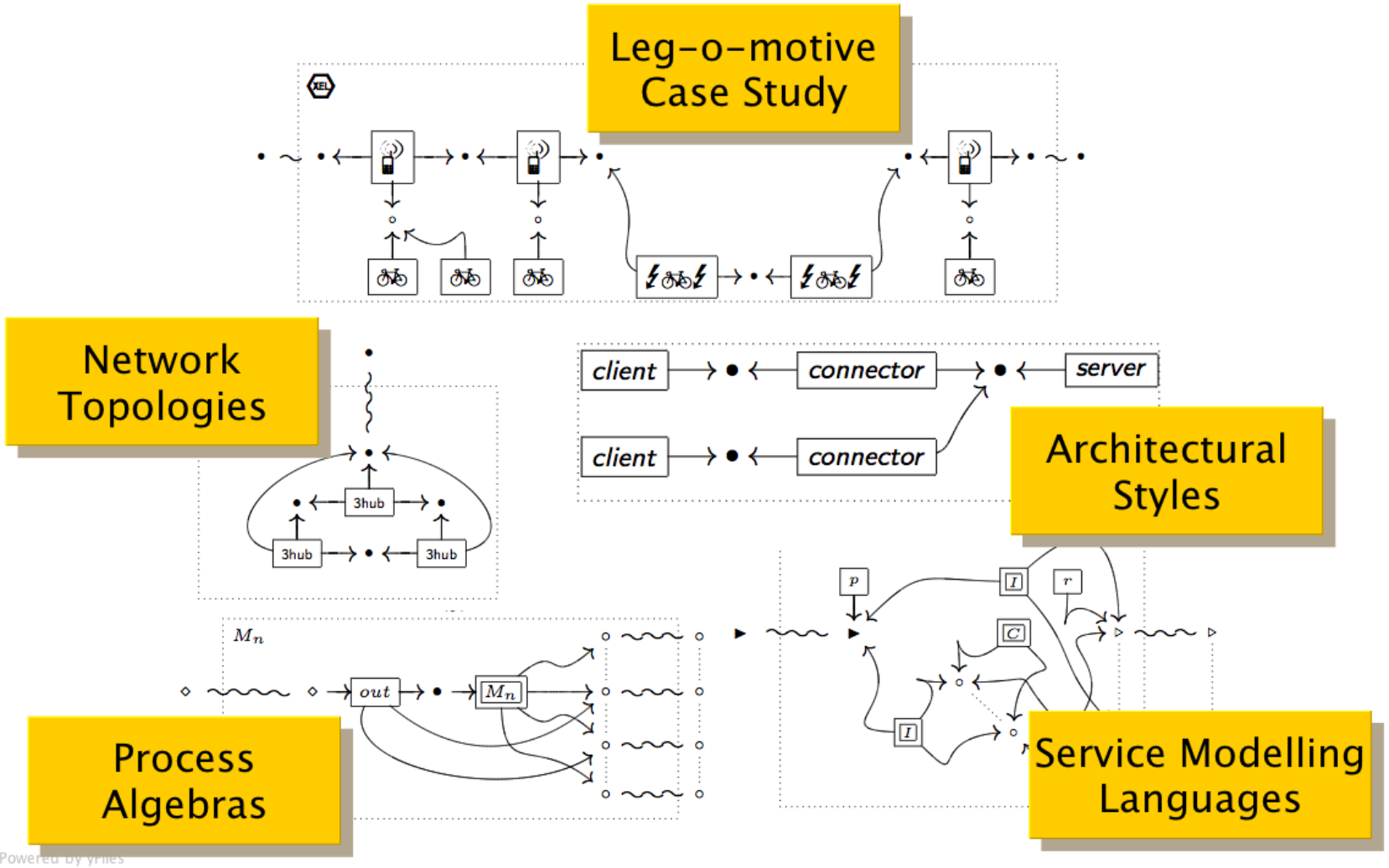


Read ADR as

- A = well formedness.
- D = diagrams.
- R = composition.

- *More in [TGC'07, D5.3b]*

More Examples



Summary and Conclusion

What is ADR?

- **A**lgebra of architectural **D**esigns that can be **R**econfigured.
- Based on term rewriting, (hierarchical) graphs.

What can I do ADR?

- Build consistent architectures.
- Reconfigure architectures.
- Analyse architectures.

Pointers

More on ADR can be found by

- Going to `http://www.albertolluch.com/adr.html`
- Reading
  - *Hierarchical Design Rewriting* (WRLA'08)
  - *Service Oriented Architectural Design* (TGC'07)
  - *D5.3b Requirements for automated reconfiguration and specification of policy run-time support*
  - *Style-Based Architectural Reconfigurations* (EATCS)
- Contacting us.
- Pay attention to Roberto's talk.