ADR at Work, part I (Past)

Alberto Lluch Lafuente\textsuperscript{1} (speaker)
Roberto Bruni\textsuperscript{1}, Ugo Montanari\textsuperscript{1}, Emilio Tuosto\textsuperscript{2} (contributors)

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

Sensoria Workshop München, March 11-14, 2008
ADR is a three-letter acronym that may refer to:

- Académie de Roberval, a school in Montreal, Canada
- short for Accord européen relatif au transport international des marchandises dangereuses par route, 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 $\phi$.
  - **P1.SOC**: applications have holes (services).

- **P2**: Reconfigure architectures preserving $\phi$.
  - **P2.SOC**: Holes (services) to be reconfigured internally.
Problem statement

Main problems ADR faces:

- **P1 := Build architectures with structural properties $\phi$.**
  - P1.SOC := applications have holes (services).

- **P2 := Reconfigure architectures preserving $\phi$.**
  - P2.SOC := Holes (services) to be reconfigured internally.

Some flaws of existing approaches

- **P1 via ...** Drop&Bind ingredients, check $\phi$: tedious.
- **P1 via ...** Bounded SAT: no guidance, trial&error.
- **P2 via ...** Show $\phi$-preservation: manual.
- **P2 via ...** Monitor $\phi$-preservation: no guarantee.
Principles of ADR

**Architectural Design Rewriting?**

- **Algebra of designs**
  - Type $T_\phi$ set of architectures that satisfy $\phi \leadsto P1$.
  - Set of design productions (operations, inductive definitions).

- **Domain**
  - Designs: graphs with interfaces.
  - Partial designs: designs with holes $\leadsto P1.SOC$.

- **Rewriting**
  - Rewrite design terms (not designs) $d : T \rightarrow d' : T \leadsto P2$.
  - Based on conditional term rewriting, SOS $\leadsto P2.SOC$. 
Principles of ADR

**Architectural Design Rewriting?**

- **Algebra of designs**
  - Type $T_\phi$ set of architectures that satisfy $\phi \sim P1$.
  - Set of design productions (operations, inductive definitions).

- **Domain**
  - Designs: graphs with interfaces.
  - Partial designs: designs with holes $\sim P1.SOC$.

- **Rewriting**
  - Rewrite design terms (not designs) $d : T \rightarrow d' : T \sim P2$.
  - Based on conditional term rewriting, SOS $\sim P2.SOC$.

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)

- **filter** : \( \rightarrow \) FILTER

- **bypass** : \( \rightarrow \) FILTER

- **par** : FILTER FILTER \( \rightarrow \) FILTER

- **seq** : FILTER FILTER \( \rightarrow \) FILTER

Powered by yFiles
Ex1: Pipes-and-Filters (D)

\[ \text{seq}(\text{filter}_1, \text{par}(\text{filter}_2, \text{filter}_3)) \]
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 $\phi$ 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 \( \phi \) 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 \( \psi \).

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

We require some ordering constraints $\phi$ 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 $\psi$.

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

We ask for an architecture satisfying $\phi$ and preserving $\psi$.

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

- Leg-o-motive Case Study
- Network Topologies
- Architectural Styles
- Process Algebras
- Service Modelling Languages
Summary and Conclusion

What is ADR?
- Algebra of architectural Designs that can be Reconfigured.
- 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.