OOP and XML

Alain Frisch

http://www.cduce.org/
Département d'Informatique
École Normale Supérieure

ECOOP Panel, 2004-06-17

" *The essence of XML:*

- *The problem it solves is not hard.*
- *It doesn't solve it very well.* "

P. Wadler (*The Essence of XML* - POPL 2003)

XML and OO: common interests

	OO	XML
Extensibility	Subclassing	New tags
Interfaces	Method prototypes	Schemas
Modularity	Classes	Namespaces
Componentization	Classes	Structural types
Self-description	RT class info, inspection	Tag names

Antogonistic points of view

Antagonistic: adj, incapable of harmonious association.

XML

XML is all about data.

- Data-model
- Expose data
- Types for data
- Global+transparent view of data
- Data exist before operations

OO

OO is all about hiding data.

- Programming paradigm
- Hide/encapsulate data
- Types for operations
- Local+opaque view of data
- Data is just the place where operations occur

"XML is not OO."

(J.C. Clark, PLAN-X 2002)

What is XML good for ?

Good at nothing, but intended for:

- storing data (that will outlive applications).
- exchanging data between unrelated applications.
- representing data within an application ?

Programming language support for XML ?

Why ?

Serialization

- XML as a persistent representation of application data ?
- Just a concrete representation for internal data structures.
- Issue: robustness w.r.t. evolution of the application.
- Not really interesting.

Free the data !

- Componentization of applications *around* XML data (many small modules working on common data).
- Data-centric applications (query, transformation).

Combining OO and XML ?

- Combine = “merge” ?
- Combine = “use together” ?

Combining OO and XML ?

It's ok to use OO and XML together, which may require to merge *some* concepts of OO and XML.

- E.g.: type system that includes classes, basic data types, and XML schemas.

It's wrong to unify OO and XML data models, because they are antagonistic.
Data-binding approaches kill the benefits of XML.

Statements

For data-centric applications, it's a good idea to have support for the data model in the language.
Functional programming ?

<plug kind='shameless' demo='http://cduce/'>
CDuce = a typed functional language, with built-in support for XML (types, patterns, iterators), first-class+overloaded functions, dynamic dispatch on types, ...
</plug>

If XML is just an concrete representation of application data, the application doesn't need the full flexibility of XML and XML types. No special support in the langage, just write I/O functions by hand (or not).

Issues, questions

- Combined data model.
- Foundation of type systems, subtyping, ...
- Structural vs. named types.
- Extending XML / extending classes.
- XML manipulation paradigms (pattern matching, paths, queries, filters, iteration, navigation). Can they benefit to "standard" objects ?
- Algorithmic issues (dispatch on XML types, efficient functional manipulation of XML).