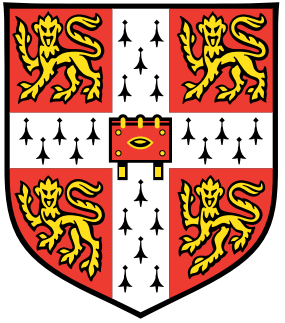
# XML in Siesta

## - background and usage

Toby White

Dept. Earth Sciences,
University of Cambridge



```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet href="http://cmlcomp.org/ccViz/display.xsl"
 type="text/xml"?>
<cml xmlns="http://www.xml-cml.org/schema"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:dc="http://purl.org/dc/elements/1.1/title"
 xmlns:siesta="http://www.uam.es/siesta/namespace"
 xmlns:siestaUnits="http://www.uam.es/siesta/namespace/units"
 xmlns:eMinerals="http://www.eminerals.org/namespace">
 <metadata name="eMinerals:cmlSubsetVersion" content="0.9" />
 <metadata name="dc:contributor" content="xmlf90 v1.99" />
 <metadataList>
  <metadata name="siesta:Program" content="siesta" />
  <metadata name="siesta:Version" content="siesta-2.0-release" />
  <metadata name="siesta:Arch" content="i686-pc-linux-gnu--Nag" />
  <metadata name="siesta:Flags" content="f95 -g -mismatch -kind=byte -dcfuns" />
  <metadata name="siesta:StartTime" content="2006-03-20T16-47-21" />
  <metadata name="siesta:Mode" content="Serial" />
  <metadata name="siesta:Nodes" content="1" />
  <metadata name="siesta:NetCDF" content="false" />
 </metadataList>
 <parameterList title="Input Parameters">
  <parameter name="SystemName" dictRef="siesta:sname"
   value="2,2&amp;quot;-dichloro-biphenyl" />
  <parameter name="SystemLabel" dictRef="siesta:slabel" value="22PCB_torsion" />
  <parameter name="LongOutput" dictRef="siesta:verbosity" value="false" />
  <parameter name="NumberOfSpecies" dictRef="siesta:ns" value="3" />
```

# CML = Chemical Markup Language

- provides structures for chemically-relevant data

- in existence since 1996

- motivated by opaque data handling in crystallography

# CMLComp = CML for Computation

- adapt CML for computational atomistics

- in development since 2002

- SIESTA is flagship code

# CMLComp

currently available

SIESTA
GULP
DL_POLY

code complete, in next release

CASTEP

under development

MOPAC
GAMESS-US
GAMESS-UK

## What is CMLComp?

sort of a data format...

but not really, since data formats are annoying and fragile and another thing to learn and you'd never get everyone to agree on one and anyway lots of concepts aren't shared between codes but you still want them in your CMLComp file ...

CMLComp is an *XML language*

*uses a standard, well defined syntax*

with a set of *microformats* for marking up scientific (chemical) data.

*so you can understand the data*

and a *document structure* for ordering the data

*and you know where to find it*

# Microformats
# - a (simplified) example

(Siesta)

```
<property title="Kohn-Sham energy">
  <scalar dataType="real" units="eV">
    -35468.83
  </scalar>
</property>
```

(Gulp)

```
<property title="Velocity P-wave (Voigt)">
  <scalar dataType="real" units="km/s">
    38.14290781146
  </scalar>
</property>
```

# Document Structure

*(like writing up a practical at school!!)*

| | | |
|---|---|---|
| metadataList | | `<metadataList>` |
| parameterList | | `<parameterList>` |
| module | (initial state) | `<module title="Initial System">` |
| module | (step 1 ... | `<module role="step" index="1">` |
| module | ... step n) | `<module role="step" index="100">` |
| module | (final state) | `<module title="Final System">` |

Microformats + Document Structure

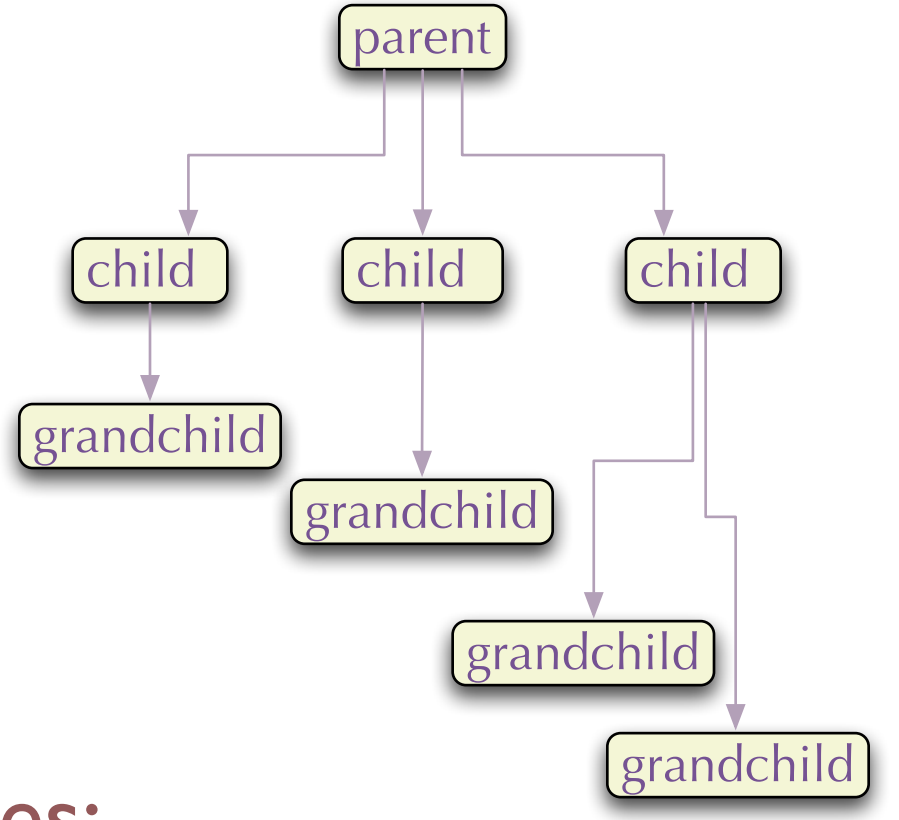
⇒ Agnostic tools

*(eg ccViz)*

XML ⇒ available toolkits

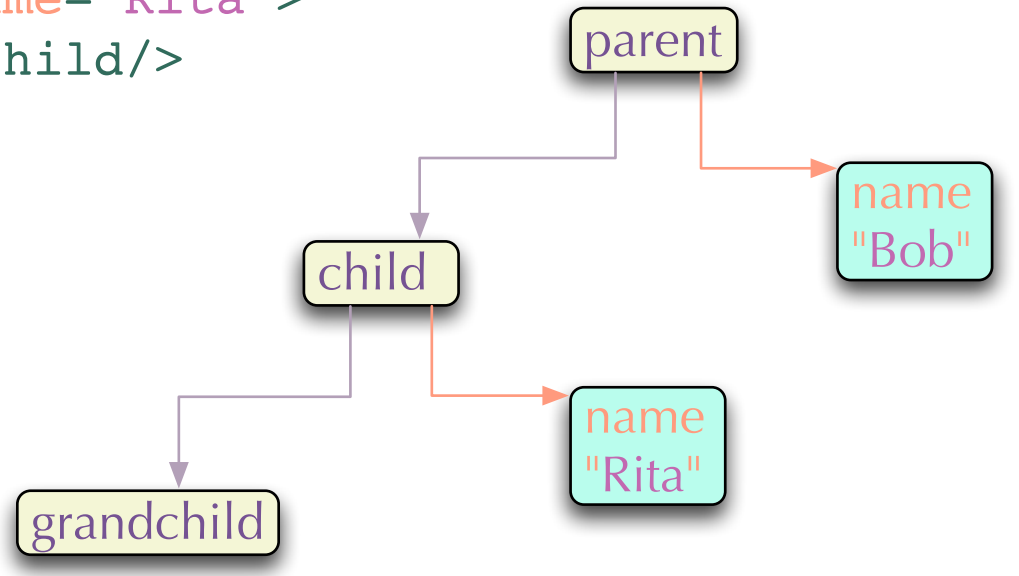
*(build tools easily in any language)*

```
<parent>
  <child>
    <grandchild/>
  </child>
  <child>
    <grandchild/>
  </child>
  <child>
    <grandchild/>
    <grandchild/>
  </child>
</parent>
```



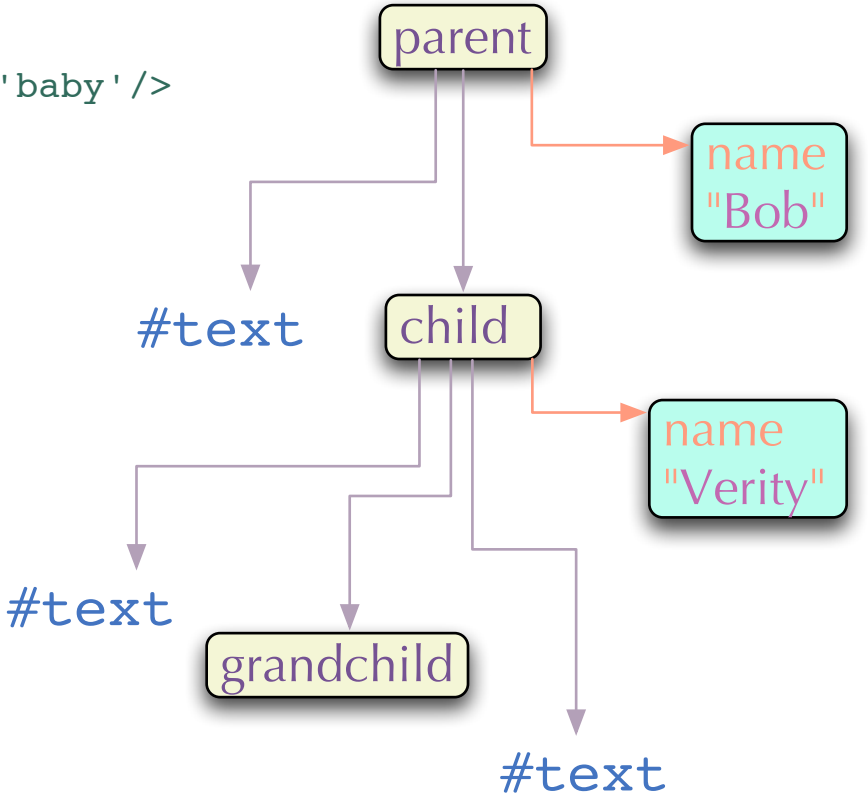

# XML trees: Elements

```
<parent name="Bob">
  <child name="Rita">
    <grandchild/>
  <child>
</parent>
```

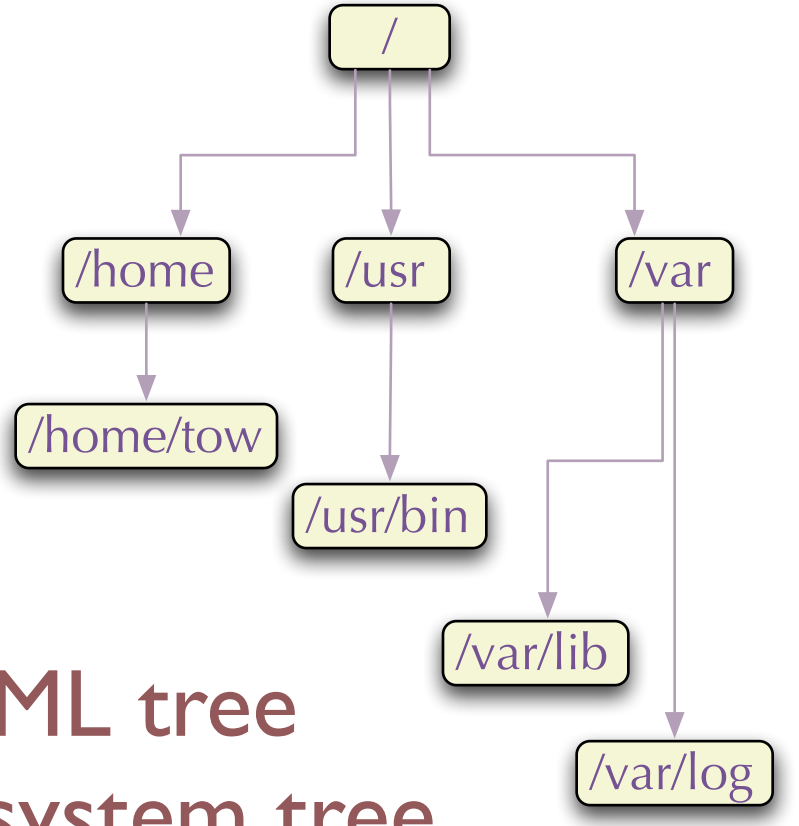



# XML trees:
# attribute names and values

```
<parent name="Bob">
  Bob is a doctor
  <child name="Verity" nickname="Rita">
  Rita is a lawyer.
    <grandchild name=''
                nickname='baby'/>
  Rita is Bob's child.
  </child>
</parent>
```



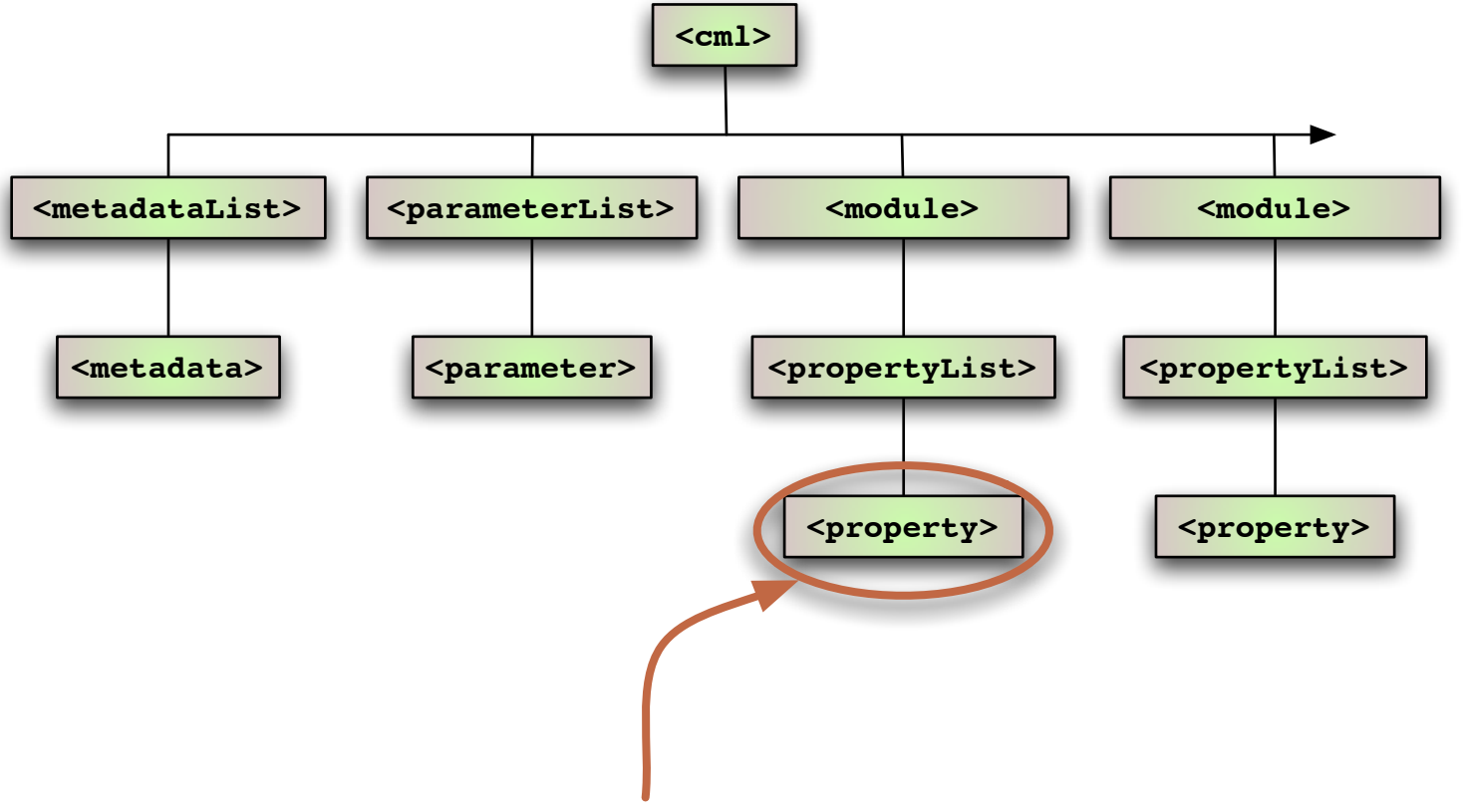

# XML trees: text




XML tree

≈ filesystem tree

**XPath**




/cml/module/propertyList/property

Extract all input parameters:

```
/cml/parameterList/*
```

Extract everything calculated in the last step:

```
/cml/module[last()]/*
```

# Dictionaries

An un-simplified microformat:

```
<property title="Kohn-Sham energy"
  dictRef="siesta:E_KS">
  <scalar dataType="real"
    units="siestaUnits:eV">
-35468.83</scalar>
</property>
```

# Dictionaries

define

identify

disambiguate

*... platform for conceptual resolution ...*

... actually, this will be easier if I wave my hands about ...

# Generating CMLComp:

FOX http://www.uszla.me.uk/FoX/

Fortran library for XML

*simple CML output for non-computer scientists...*

```
call cmlAddMolecule(coords, labels)
call cmlAddProperty(name, value, &
                    dictRef, units)
             etc...
```