Extensible markup language (XML) can be used for flexible data representation, GUI data input via a browser, input validation, native database storage with read/write search access. A W3C standard that along with HTML/WebDAV gives universal access. Compatible with legacy codes.
Everything is data

- The application interfaces (XForms), the class definitions or schema (XSD), input and output processing (XSLT), metadata, input namelists.
- All can be stored in a native XML database (eXist or XML BDB.)
- XForms is a W3C standard.
  - In the process of being implemented.
  - Server side (Orbeon): slower, but don't need plugin.
  - Client side (Mozilla XForms): very fast, most advanced, user has to install.
  - Eventually will be integrated into browsers (eg SVG).
User only needs a browser

URI to XForm, static url or database query

XForm retrieves template instance

XForm stores new input, can be run from monitor component
XForms example

Source XML and XSD (not shown)
<parameter><name>aconc</name><value type="xsi:float">1.0</value>
<help>Ion concentration array size nspec</help>
</parameter>
<parameter><name>sp_xs1</name><value>0.0</value>
<help></help>
</parameter>
<parameter><name>sp_xs1</name><value>0.0</value>
<help></help>
</parameter>
</namelists>

Source XForm ~100 lines
<table xf:repeat-nodeset="instance('namelists')/namelist">
<thead><tr><th>Namelist: xf:output ref="@name"/></th></tr></thead>
<tbody><tr>
<td>xf:output ref="name">
<xf:hint ref="../help"></xf:hint></td>
<td><xf:input ref="value"></xf:input></td>
<td><xf:output ref="units"></xf:output></td>
</tr>
</tbody></table>
**XSLT: XML->NML**

Advantages of XML with legacy compatibility

```
&toric_mode
toricmode=toric
/
&toricinp
isol=1
nvrb=3
nelm=230
ntt=512
nmod=255
nptvac=-1
mxmvac=15
freqcy=80.0E6
nphi=10
Antlen=48.0
...`

```
<xsl:template match="/">
  <html>
    <body>
      <pre><xsl:text>&#xA;</xsl:text>
        <xsl:for-each select="namelists/namelist">
          &amp;<xsl:value-of select="@name"/>&#xA;
          <xsl:for-each select="parameter">
            <xsl:text>=</xsl:text><xsl:value-of select="value"/>
            <xsl:text>&#xA;</xsl:text>
          </xsl:for-each>
        </xsl:for-each>
      </pre>
    </body>
  </html>
</xsl:template>
```
Making it easier

- As a general principle, users do not want to and should not edit source XML.
- Can use an XForm to generate XML or input from Fortran namelist or a custom flat file to generate initial template that is transformed to XML via XSLT:

```plaintext
#First namelist
""
example_namelist1
""
#name, #type,       #value,       #units, #help
idebug,  logical,  1,            ,      , "Turn debugging on."
bzero,   float,    4.3,          "Tesla", "Vacuum toroidal field on geometric axis."
x,       integer,  122,          ,      , "Number of mesh points in x direction."
runmode, string,  "diagmode",    ,      , "Running mode. One of solmode, diagmode."

#Second namelist
```
Summary

- Use a data-centric model for input creation (code namelists, IPS config file, etc)
- Database provides repository for input templates, interfaces, and conceivably results metadata
- Automatically get some provenance information, queries (Xpath or Xquery), other database functions.
- XForms provides through XSD some degree of input validation
- Future: we've dealt with the nodes, what about the edges – semantic relation validation (do my inputs even if in valid ranges actually form a valid consistent set?)