TEI, a new phase

The P5 release of the TEI Guidelines has three aims:

Interoperability taking advantage of the work done by others
Expansion addressing areas as yet untamed
Internal audit cleaning up the accretions of a decade

... all without losing touch with its core constituency
Interoperability

A lot of other people have been working in this area since 1987!
TEI P5 must fit into a joined-up digital world, along with

- W3C standards (XLink, schema, etc)
- Unicode character encoding
- Specialized markup vocabularies (MathML, SVG, DocBook, etc)
- Other metadata schemas (METS, EAD, etc)
- Other conceptual models and ontologies
- .... and TEI P4
Expansion: why?

- TEI P4 did not (could not) cover everything!
- The TEI has always been ahead of the pack in promoting evolutionary change:
  - Some parts of TEI P4 were successfully experimental (e.g. the extended pointer syntax, corpus metadata)...
  - ... some were influentially experimental and have become FaQs (‘frequently answered questions’) e.g. synchronization and standoff
  - ... others were just experimental, and have been overtaken by events (e.g. writing system declaration, feature structures, terminology...)
- A key deliverable: better tools for customization and integration
Internal audit: how?

The TEI toolkit:
- an XML editor
- a library of XSLT scripts
- a real version control system
- test suite

Working practices:
- the workgroup model
- role of the council
- release cycles

Opening the TEI
Major "messages" about P5

- Customizability
- Modularity
- Internationalization
- New coverage
The TEI Guidelines, its DTD, and its schema fragments, are all produced from a single XML resource containing:

1. Descriptive prose (lots of it)
2. Examples of usage (plenty)
3. Formal declarations for components of the TEI Abstract Model:
   - elements and attributes
   - modules
   - classes and macros
4. We call this resource an ODD (One Document Does it all) although the master source is instantiated as a gazillion XML mini-documents.
The TEI scheme can only be used by customizing it. Customizations are also expressed in the ODD language.

For example:

```xml
<schemaSpec ident="myTEIlite">
  <desc>This is TEI Lite with simplified heads</desc>
  <moduleRef key="core"/>
  <moduleRef key="tei"/>
  <moduleRef key="textstructure"/>
  <moduleRef key="header"/>
  <moduleRef key="linking"/>
  <elementSpec ident="head" mode="change">
    <content><rng:ref name="model.text"/></content>
  </elementSpec>
</schemaSpec>
```

produces the schema for TEI Lite, with a slight change.
Taking our own medicine

- We use a library of XSLT scripts which can generate
  - The book in canonical TEI XML format
  - The book in HTML or PDF
  - RelaxNG, DTD, or W3C schema fragments

- The same library is used by Roma: a web-based customization tool which can generate
  - project-specific documentation
  - project-specific schemas
  - translations into other (human) languages
Modularity

- Uniformity of module structure (goodbye to the pizza model)
- Uniform naming scheme for classes, macros, datatypes
The TEI abstract model

- Each element declares the module it belongs to: elements cannot appear in more than one module.
- A markup scheme (a schema) consists of a number of discrete modules, which can be combined more or less as required.
- A schema is made by combining references to modules with other declarations.
- Each module extends the range of elements and attributes available by adding new members to existing classes of elements.
The rise of the class system (1)

- Class membership can do two distinct things for an element:
  1. give it some attributes
  2. allow it to join a ‘club’

- Content models reference ‘clubs’ rather than specific elements (wherever possible)

- There are two kinds of club:
  - model.xxxLike  sibling elements which are semantically alike
  - model.xxxPart  sibling elements which constitute another one
The rise of the class system (2)

- Classes are easier to understand and remember than elements
- Adding a new element becomes a matter of deciding what it is ‘like’, or what it is a ‘part’ of
- Specialization of the TEI generic structure for specific needs is a simple declarative matter
Why the stress on customization?

The TEI has over 20 modules. A working project will:

- Choose the modules they need
- Probably narrow the set of elements within a module
- Probably add local datatype constraints
- Possibly add new elements
- Possibly localize the names of elements

We can do all that in an ODD
All TEI elements are surrounded by a naming layer, which allows their user-visible names to be changed. This covers:

- element names
- attribute names
- attribute values
- short descriptions

The translation database is maintained separately, so attribute names and values are translated once only; but all descriptions etc. are stored in the same ODD source.
Our gesture towards ontological mapping

The `<equiv>` element supplies a URI which identifies an equivalent concept (*not* a name) in some externally-defined ontology, e.g.

- ISO data category registry
- CIDOC conceptual reference model
- Wordnet

It can also be used to specify a stylesheet transformation where syntactic sugar has been applied, for example to specify formally that `<placeName>` is equivalent to `<name type="place">`
New content in TEI P5

- authoring and **tag documentation**
- manuscript description
- `<choice>` and `<combine>`
- prosopographic elements (in progress)
- character and glyph documentation beyond Unicode
- linking methods
- feature structures
- class system and core elements overhaul
- structured bibliographic elements
- dictionaries and termbanks

And some things may be ruthlessly excised...
Here be Dragons!

Will my old files work with P5?

- `<TEI.2>` is now `<TEI>` and `<teiCorpus.2>` is now `<teiCorpus>`
- TEI elements are in the http://www.tei-c.org/P5/ namespace
- many attribute values seem to have changed... Y|N is now true|false
- ... or disappeared
  - text-valued attributes have become child elements
  - `lang` has become `xml:lang`
  - `id` has become `xml:id`
  - all `IDREF` (target) values have become URLs
  - the external pointing elements seem to have disappeared!
- The TEI pizza model has been cast aside and my extension files no longer work!

Why should I bother to switch?
Here be Treasures!

Some new technical advantages:

1. Unicode
2. Better schema tools e.g. extension via namespaces
3. Better modularization tools: ODD genuinely does everything
4. Better integration with W3C standards (e.g., linking)
5. Simpler, more consistent, data model

New/improved coverage or expressiveness

1. Manuscript description
2. Feature structures
3. Dictionaries
4. Prosopography and ontologies

(And P4 will continue to be supported for five years... )
Open TEI

- The TEI consortium now releases the Guidelines under a GNU Public license
- All development now takes place in public using CVS on Sourceforge
- Feature requests and bug tracking are also on Sourceforge
- TEI components are available as Debian Linux packages

However, the name TEI remain a trademark, and technical work continues to be authorized by TEI Technical Council, elected by members of the Consortium.
Open TEI: what does it mean?

- The TEI remains a community initiative, driven by the needs of its members and users.
- To encourage more devolved development we need to build a larger community of developers.
- This means both making entry level development easier and peer approval more visible.
- Which means we need more participation from all potential TEI users, as members of SIGs, Workgroups, and Council ...