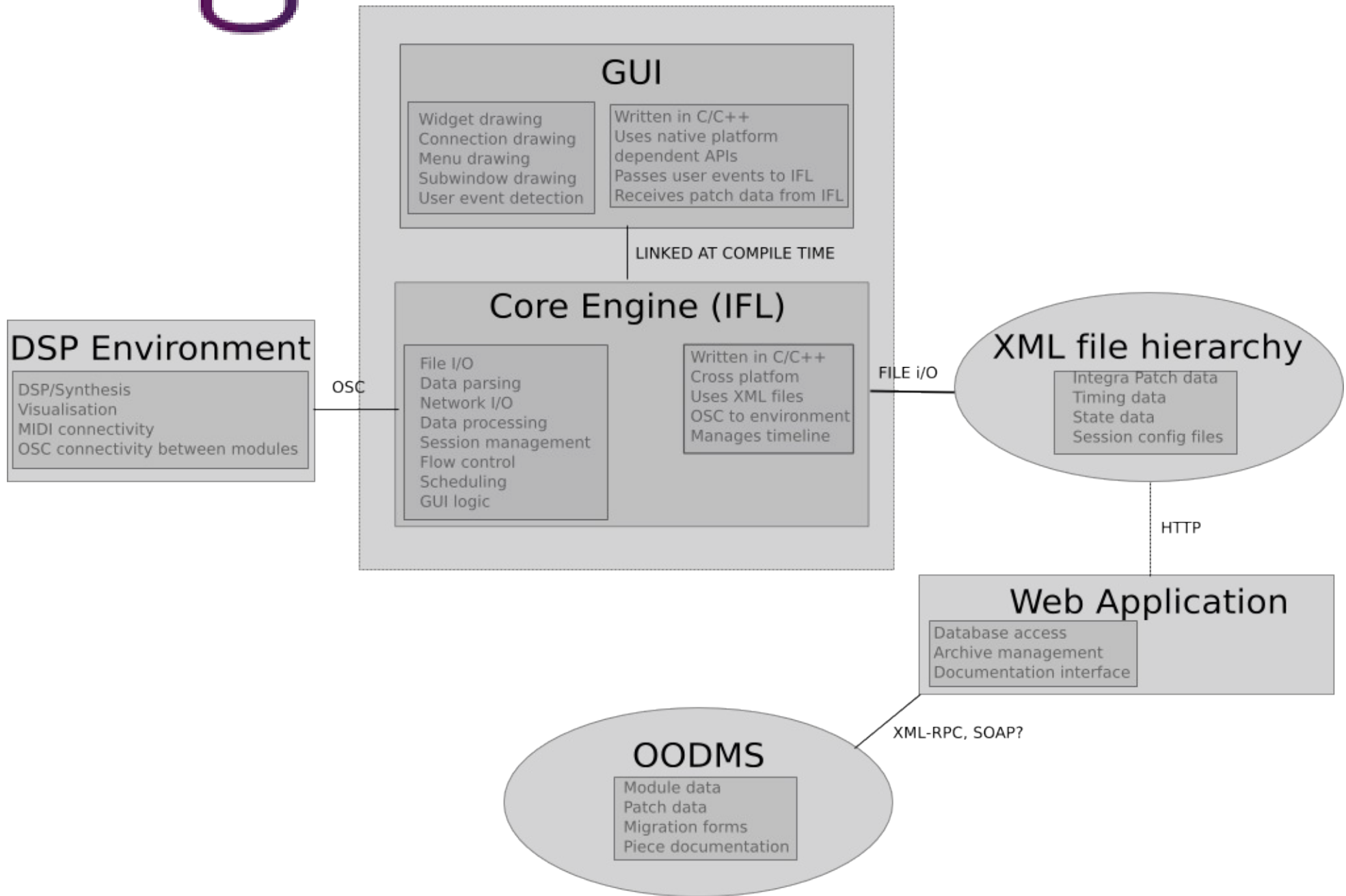




Presentation on findings resulting from
research into database technologies for the
Integra project

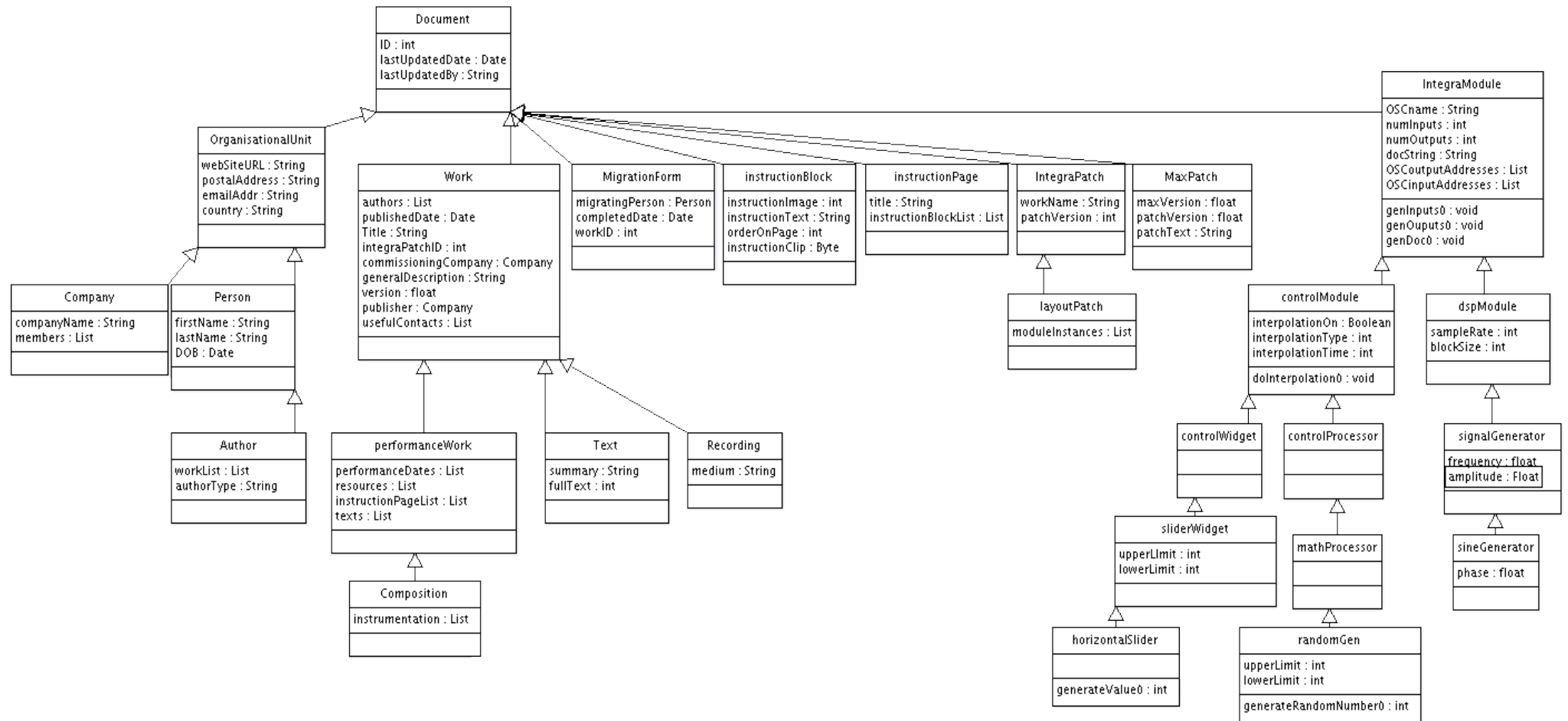
Integra

Context: Overall System



Integra

UML Class Diagram





What do we need?

- Something to store data in
- Some means of publishing that data on the web
- Some means of publishing that data in XML

Integra

Existing Approaches

Mustica

- Slick interface
- Looks like PHP (+js) + MYSQL
- CC no-derivs license
- Difficult to integrate module data

Integra

Existing Approaches

PDRP

- Educational bias
- Looks like hand rolled html
- Probably not a sustainable approach for large numbers of works

Integra

Evaluation Criteria

Essential

- Open Source
- Reasonably mature (no <v.1.0)
- Well maintained with active community
- Well documented
- API suitable for building web interface

Integra

Evaluation Criteria

Ideal

- XML support (native or via existing API)
- OODMBS or ORDBMS
- Supports query language
- Storage format closely coupled to data format

Integra

Evaluation Criteria

If object-oriented

- Straightforward to create object classes (through schema evolution)
- Multiple inheritance?
- Native XML support
- Remote method calls via xml-rpc or SOAP
- DOM support

Integra

Considerations

- MVC frameworks
- Traditional RDBMS + script approach
- OODMS + native scripting
- Hybrid approaches

MVC frameworks?

“Model-view-controller (MVC) is a software architecture that separates an application's data model, user interface, and control logic into three distinct components”



Considerations: MVC frameworks

- **Zope/Plone**

- Advantages:

- we already use it
 - Extensive features
 - Mature, good community

- Disadvantages:

- speed issues (but resolved in v 2.5)
 - Steep learning curve (but already partly mastered!)



- **Ruby on Rails**

- Advantages:

- 'elegant' design (Ruby)
 - Reputed fast development times
 - Popular, good community

- Disadvantages:

- Uses ORM not true OODB
 - Implicit rules (too much 'magic')
 - Requires parallel framework (admin overhead)



- **TurboGears**

- Advantages:

- nicely designed (response to RoR)
 - Reputed fast development times
 - Explicit rules (no 'magic')

- Disadvantages:

- Not mature (pre v.1)
 - Requires parallel framework (admin overhead)



- **Maypole/Catalyst**

- Advantages:

- nicely designed (response to RoR)
 - elegant
 - can use the oryx ORM

- Disadvantages:

- Not so mature
 - Requires parallel framework (admin overhead)

Integra

Considerations: Web development frameworks

See also Django, Quixote and many more....



Considerations: NXD

- **What is an native XML database?**

“The internal model is based on XML and the fundamental unit of storage is an XML document.” (Wikipedia)

- **XML-enabled?**

“These simply map all XML to a traditional database (e.g. relational).”



Considerations: NXD

- **Why NXD?**

Internal representation of data is closely coupled to desired output.

Less 'glue code'

Document-centric or data-centric

XML can be queried and indexed natively (XQuery)



Candidates: NXD

- Sedna (C)– Very nice. C, Python, Scheme APIs. Xquery. Omnimark 8 (commercial content processing platform)
- dbXML – Pre-production.
- eXist (Java)– Very nice. Xupdate, Xquery. Zope integration
- 4suite (Python) – Nice. Xquery. Full suite of XML processing tools
- Xindice – Poor features (XPath for query)
- BerkelyDB XML – Good but embedded.



•What is Xquery?

“XQuery is a query language (with some programming language features) that is designed to query collections of XML data. It is semantically similar to SQL.”

Integra

Candidates: OODBMS

- Ozone (Java)– Primarily a persistence layer for Java
- ZODB– No query language. Bloated.
- GOODS - Doesn't seem maintained
- MyOODB (Java) – Small, fast. Includes web development components
- Schevo(Python) – Good, focus on schema evolution. Non -standard query language



Candidates: ORDBMS

- SQLObject (Python)– ORM for SQL databases (used in turbogears). Clean -simple. Supports single inheritance.
- Active Record (Ruby) – used in RoR
- Castor/Cayenne (Java)
- Class::DBI (Perl)
- Many many more!



Candidates: Persistence Layer

- Durus (Python)
- Webware (Python) – complete web dev framework (pre v. 1.0)
- Oryx (perl) – Very nice meta-model driven object persistence with multiple inheritance (uses rdbms)
- Cog – Persistence layer for Python



- **Why not use a traditional RDBMS with an ORM?**

Impedance mismatch. Encapsulation and inheritance need to be 'fudged'. (But are our needs complex enough for this to be a serious problem)?

Integra

Finalists

- Catalyst: Nicest looking MVC but...
- Zope: No compelling reason not to use it as an MVC layer, however data store is lacking (ZODB bloated, one point of failure).

Integra

Finalists

- SQLObject – straightforward. Does everything we need in terms of storage. Python – easy interface with Zope as MVC layer. Also SQLOS – very good support in Zope
- eXist: Modern, lean, XML datastore. Well thought out. Can use LDAP for users and groups. Zope Integration through existDA. Many third party tools. No straightforward inheritance model – this would need to be imposed

Integra

Recommendation

Do a very simple prototype with each system (eXist, sqlobject), to establish which best suits our needs.

Integra

Bibliography

Bourret, R. (2005) *XML and Databases*. (accessed at www.rpbourret.com, 23/05/06)

Bourret, R. (2005) *Going native: Use cases for native XML databases* (accessed at www.rpbourret.com, 23/05/06)

Wikipedia!