CCPN: scientific data modeling and multi-language code generation

Rasmus Fogh

SDMIV2

December 2005

CCPN

- Introduction
- Goals and strategy
- Modeling and code generation
- The model
- APIs
- Future plans

CCPN

The CCPN Project

- Collaborative Computing Project for NMR
  - Started in 1999 (BBSRC funding)
- Unifying platform for NMR software
  - Similar to CCP4 (X-ray)
- Main goals:
  - Data standards and software integration
    - Model, subroutine libraries
    - Data modeling tools
  - Software development and distribution
  - Meetings and workshops

People

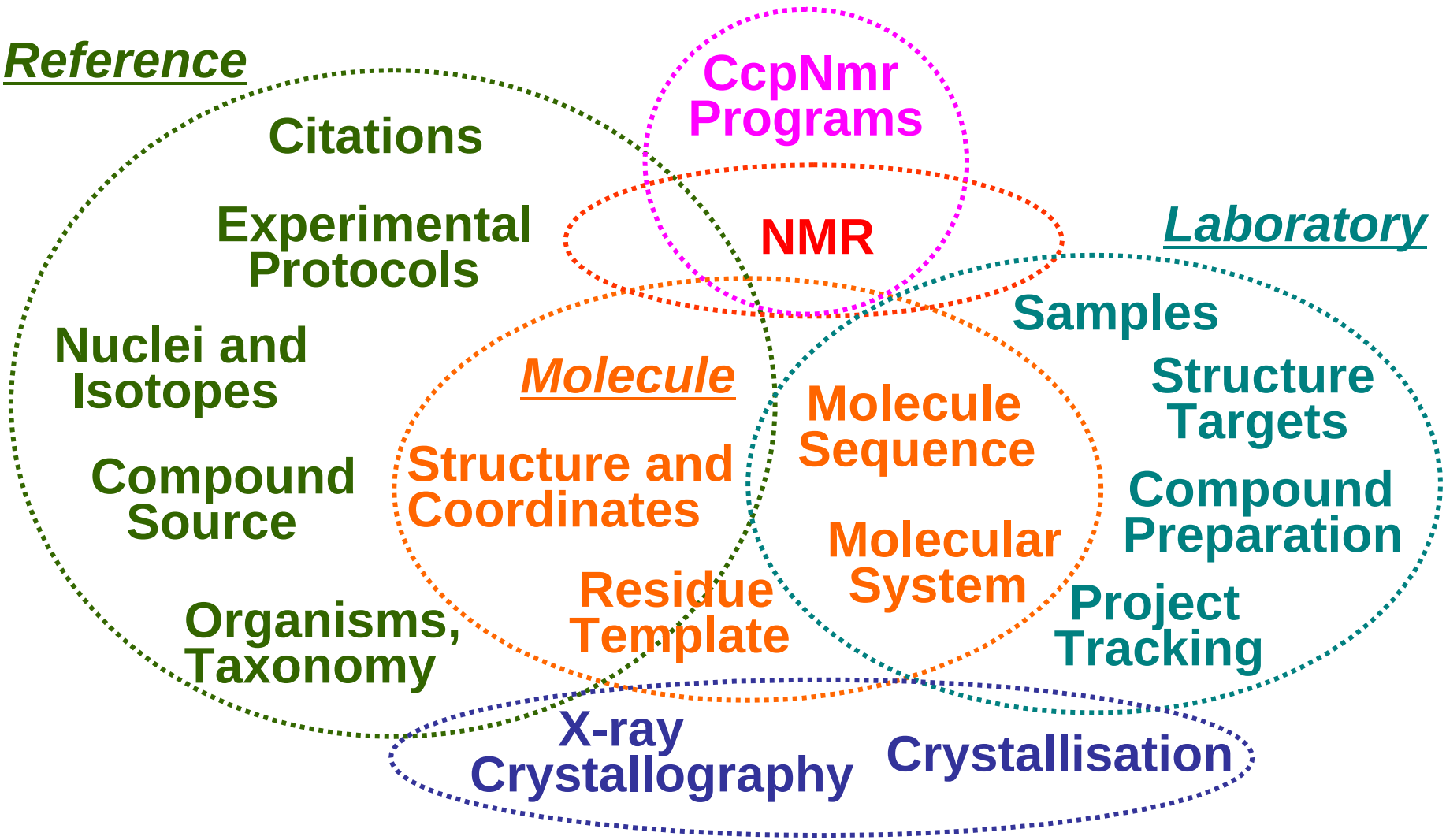
- Cambridge (Biochemistry)
  - Ernest Laue
  - Wayne Boucher
  - Rasmus Fogh
  - Tim Stevens
  - Dan O'Donovan (new!)
  - Wolfgang Rieping (new!)
- EBI (MSD), Hinxton
  - Kim Henrick
  - John Ionides
  - Wim Vranken
  - Anne Pajon
- Plus external collaborators

Credits

- Funding:
  - BBSRC (2000-2003, 2003-2006)
  - EU-NMRQUAL (2001-2004)
  - EU-TEMBLOR (2002-2005)
  - EU-NMRextend (2005-2008)
- Major Data Model contributors:
  - BioMagResBank (NMR-STAR)
  - EBI, MSD group (Molecular structure model)
  - PIMS Project (Laboratory information modeling)

CCPN

Scientific area



- Introduction
- Goals and strategy
- Modeling and code generation
- The model
- APIs
- Future plans

CCPN

NMR Software - problem

Heterogeneous development

- Lots of stand-alone programs
- Lots of proprietary data formats
  - converters necessary
- Data is ‘lost’ along the way
- Not acceptable for structural genomics projects

- No access to existing program code
- No code sharing or re-use
- You keep re-inventing the wheel
- Resources are spread thin

CCPN

Data standards

- Lossless data transfer between programs
- Completeness, integrity of data
- Data harvesting
  - all data retained till deposition
- Allows data mining

- Link and integrate software
- Work between programs
- Allow addition of modules

CCPN

Requirements

- Precise standard
- A single central description
- Validation directly against standard
- Support applications as they run.
  - Comprehensive model
  - Intermediate results
  - Consistency for rapidly changing data
- Easy to maintain and modify
- Programmer-friendly

CCPN

Standard Data Format ?

- Original approach: Specify a format

- Which to choose?
  - STAR/CIF/mmCIF
    - established, used for existing standards
  - XML
    - wave of the future
    - human readable
    - lots of software
  - SQL databases
    - heavy duty

CCPN

Standard Data Model !

- Abstract data model – UML
- No stable format – stable API
  - easier to maintain as model changes
- Support XML *and* SQL, underneath
- Support several programming languages
- How?

CCPN

Standard Data Model !

- Abstract data model – UML
- No stable format – stable API
  - easier to maintain as model changes
- Support XML *and* SQL, underneath
- Support several programming languages
- **Automatic code generation!**

CCPN

CCPN Emphasis

- Applications rather than web services
- Comprehensive storage,
  rather than message passing
- Data rather than documents
- Validity and consistency checking
- Capture data as they are created

- Introduction
- Goals and strategy
- Modeling and code generation
- The model
- APIs
- Future plans

CCPN

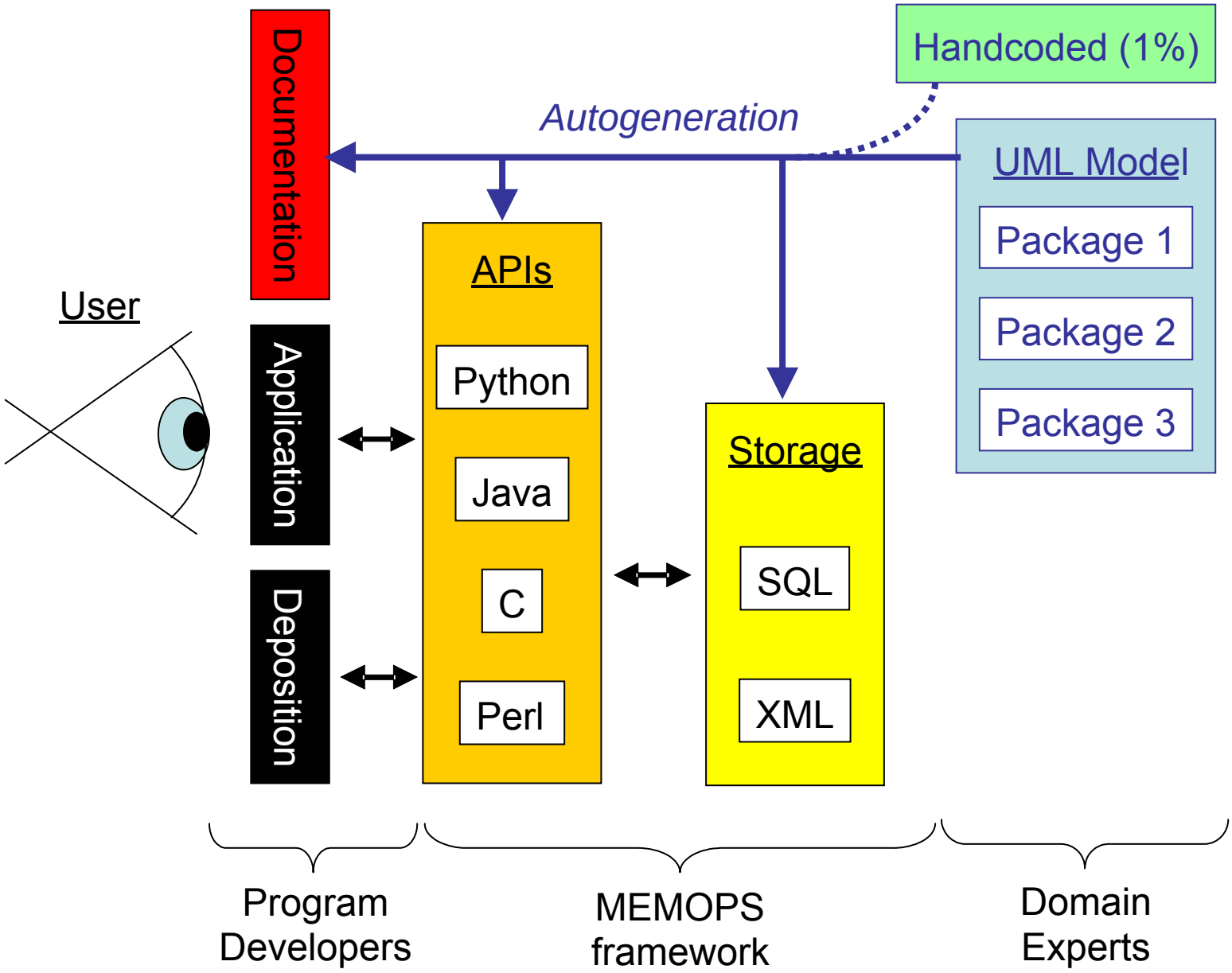
Memops modeling machinery

- Generic, content-independent
- Based on UML model
- One model –> many implementations
- Fully automatic code generation

- Implemented in Python

CCPN

Code Generation Framework



Generated code

- One single version – automatically
  - No provision for tuning
- Object oriented APIs
  - Function code derived automatically from known structure of data
- SQL and XML schemas
- I/O mappings and I/O code
- Documentation

CCPN

Code Generation Process

- 1. Edit UML model
  - *ObjectDomain*
- 2. Export model to disk
  - *CCPN Python file*
- 3. Load model into memory
  - *CCPN MetaModel objects holding information*
- 4. Generate API, I/O mappings, documentation, XML and database schemas, …
  - *Memops (CCPN) code generation*

CCPN

- Introduction
- Goals and strategy
- Modeling and code generation
- The model
- APIs
- Future plans

CCPN

Modeling Criteria

- Comprehensive
  - Contains all data you need for an application
- General
  - Can handle all likely ways of working
- Normalised
  - Data stored only once
- Unavoidably complex

CCPN

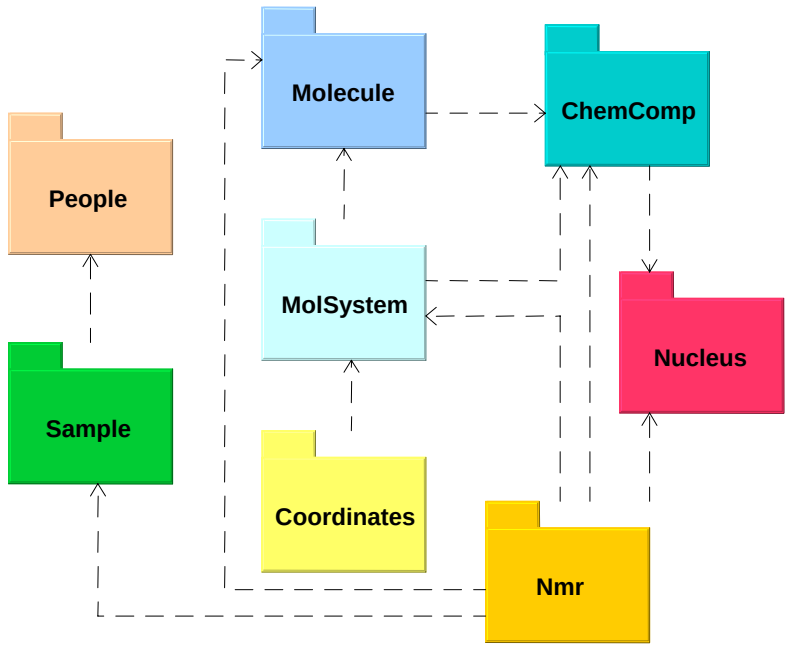
Model Contains

- Packages
  - Organises contents (model, code, and data)
- Classes
  - Attributes (e.g. strings, floats)
  - Links (connections between objects)
    - Most links are two-way
  - May be single or multiple, mandatory or optional
  - May be derived, i.e. calculated rather than stored.
  - Operations (function definitions)
    - Most are implicit

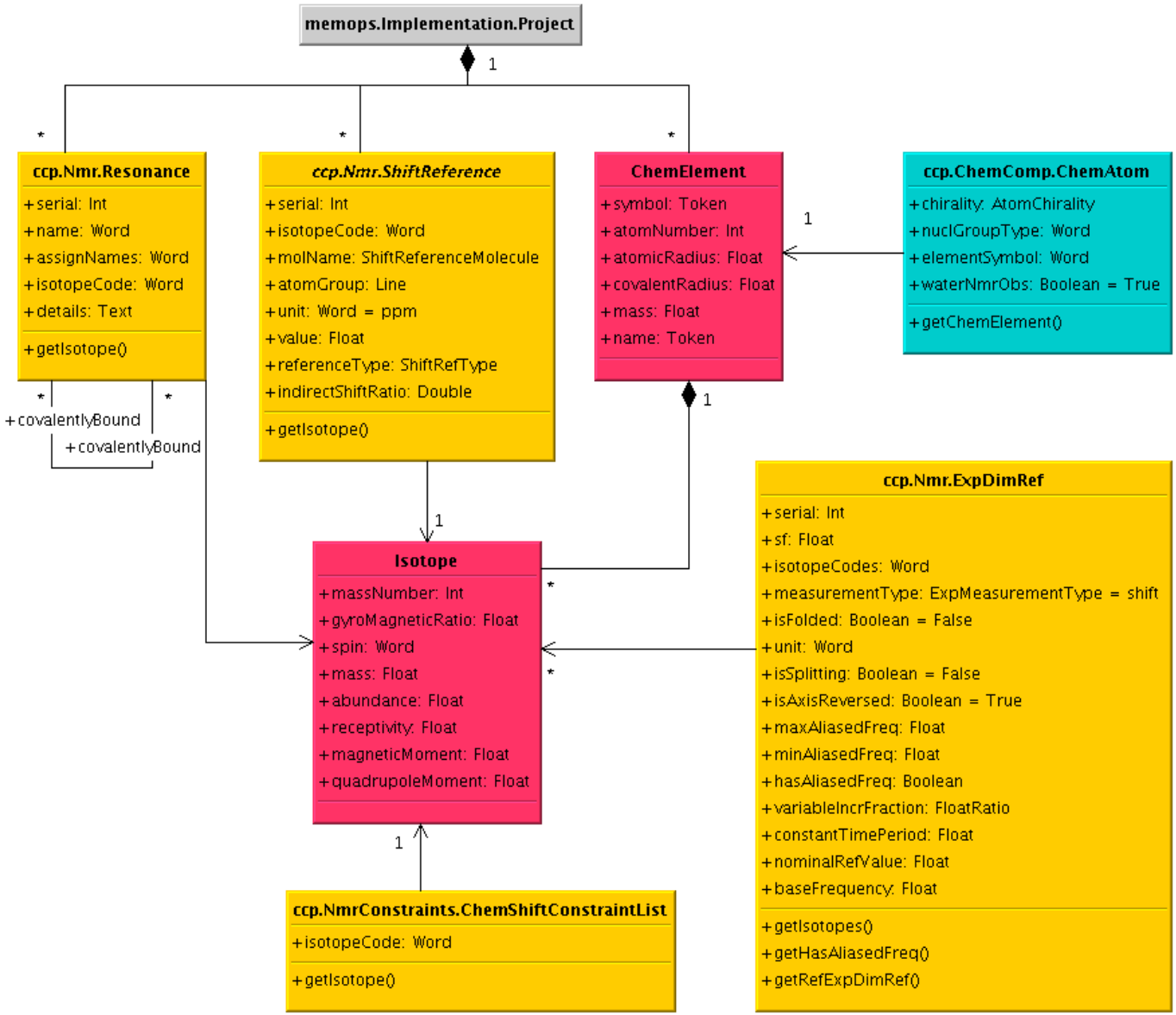
CCPN

CCPN Packages

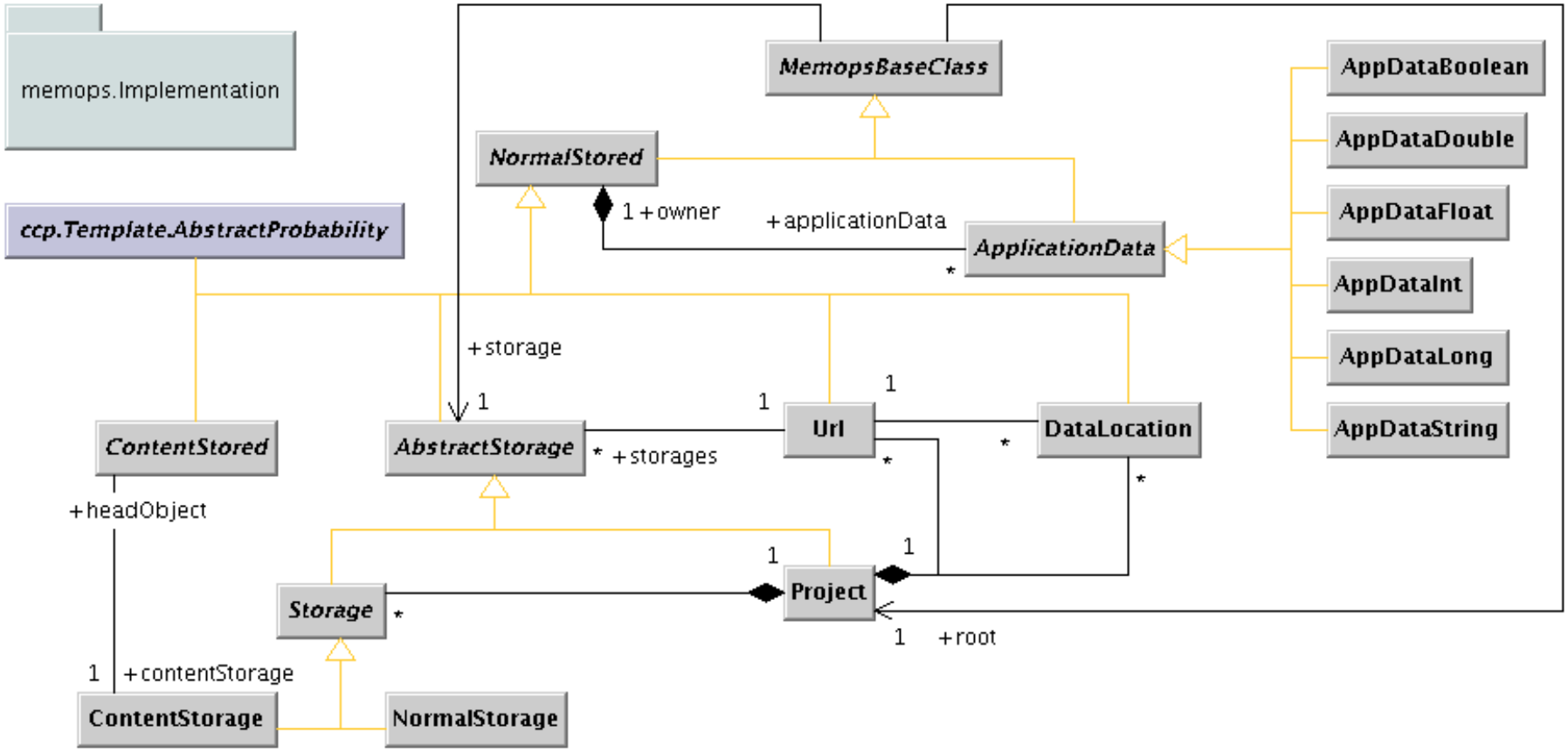
- Groupings of related data
  - e.g. NMR, X-ray, Molecular description
- Connections between packages
  - e.g. NMR loads Nucleus (isotope) information
- Allows lazy loading
  - Only load relevant data
  - Only load when a link is queried
- Save only modified
- Reference packages
  - Chemical compounds,
  - Reference chemical shifts



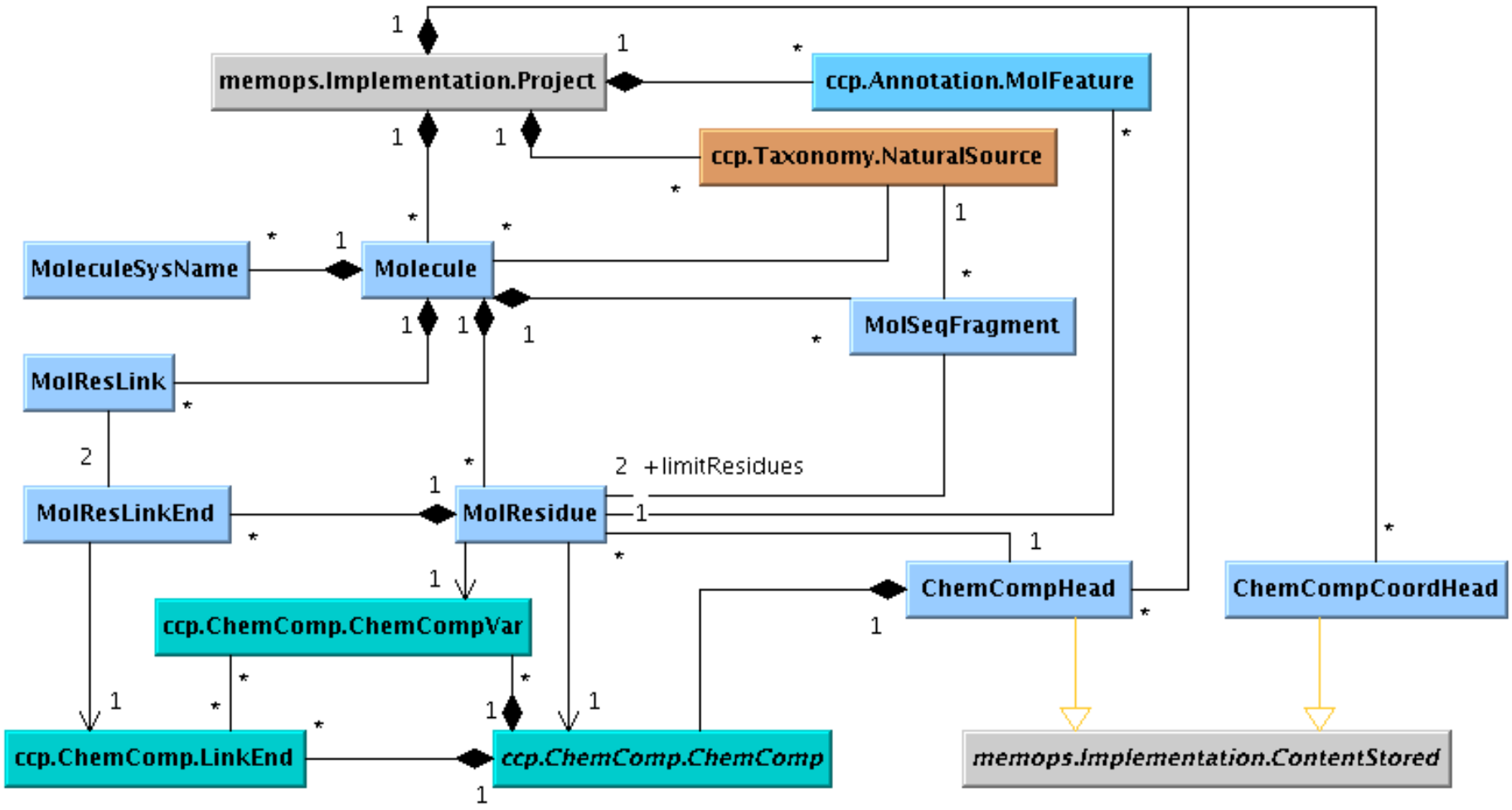
ChemElement - Details



Implementation



Molecule


memops.Implementation.Project
ccp.Annotation.MolFeature
ccp.Taxonomy.NaturalSource
MoleculeSysName
Molecule
MolSeqFragment
MolResLink
MolResLinkEnd
MolResidue
2 +limitResidues
ccp.ChemComp.ChemCompVar
ChemCompHead
ChemCompCoordHead
ccp.ChemComp.LinkEnd
ccp.ChemComp.ChemComp
memops.Implementation.ContentStored

- Introduction
- Goals and strategy
- Modeling and code generation
- The model
- APIs
- Future plans

CCPN

CCPN API Overview

- Classes for developers
  - Data access only (get, set, create delete)
  - Fully functional, complete consistency checking
    - Type, cardinality, handcoded constraints
  - Data loading handled automatically
- Currently 35 packages, 336 classes
- Autogenerated code contains:
  - 615 000 lines Python (XML)
  - 725 000 lines Python API documentation
  - 1 422 000 lines Java (XML and SQL)

CCPN

Developer Benefits -1

- Precisely specified data model and API
- No I/O code
- Validity checking
- Concentrate on science, not bookkeeping

CCPN

Developer Benefits - 2

- Extendible
  - Application data can be assigned to any object
  - UML model can be extended
    - New scientific areas
    - Custom packages
- Notification system
  - Register interest when specified attribute changes (class, not object, level). E.g. for GUIs.

CCPN

Current Status of API

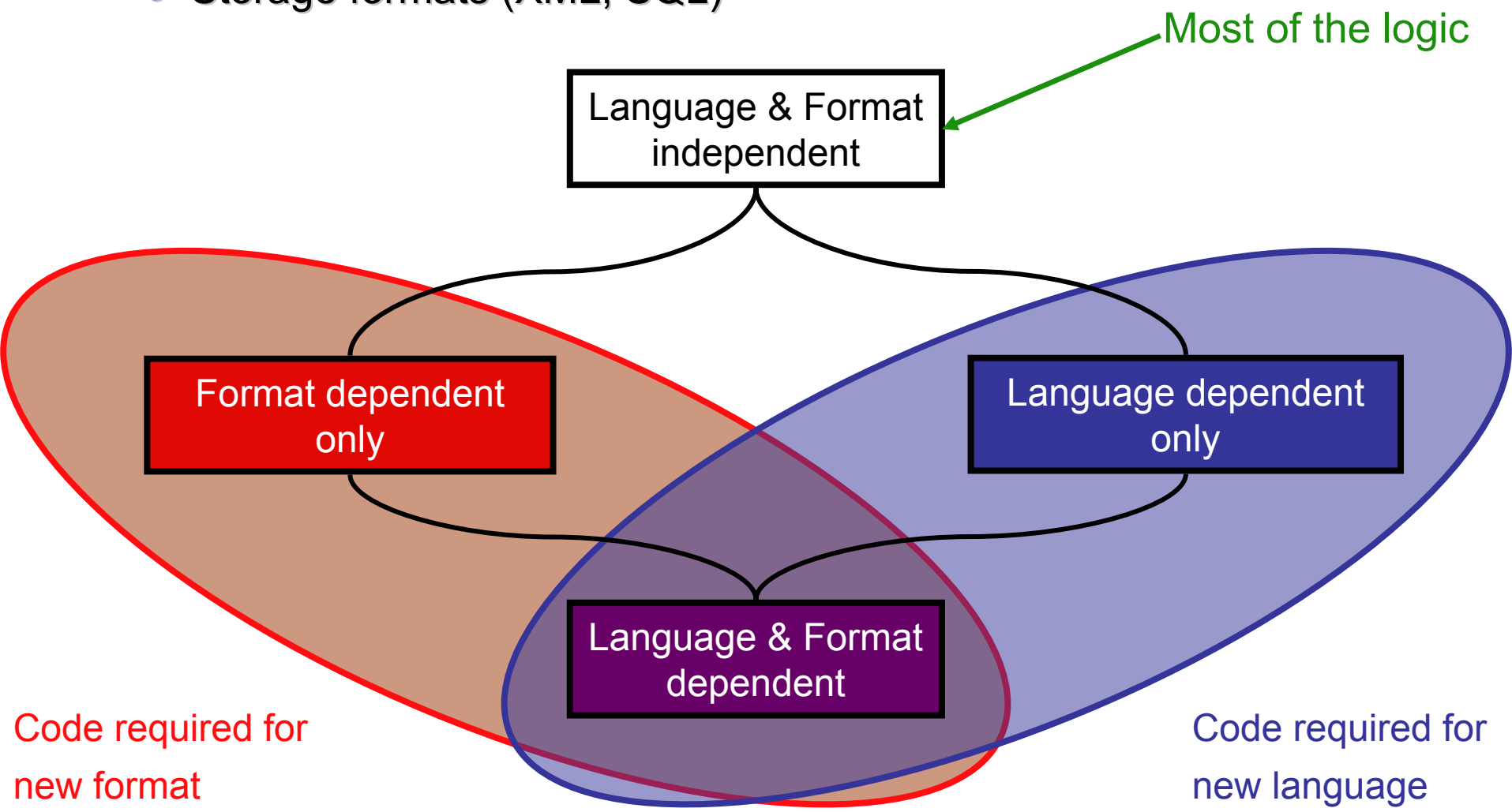
- Stable, released and tested:
  - Python and XML API and code generation
  - NMR, molecule description and structure data model
- In testing stages:
  - Java and XML/SQL API and code generation
  - Protein production data model (PIMS)

- Introduction
- Goals and strategy
- Modeling and code generation
- The model
- APIs
- Future plans

CCPN

New Core generation technology

- Reduce burden of adding new languages, formats
  - Languages (Python, Java, C++, Perl)
  - Storage formats (XML, SQL)



Plans for Languages and Formats

Language

Format	Python	Java	C/C++	(Perl)	Fortran??
XML	- Analysis - Format-Converter	- Bruker TopSpin - NMRVIEW	- Azara - Extend-NMR - NMRPIPE - AUTOPSY - (Varian) - (CYANA)	(Bioinformatics)	CYANA? Molecular dynamics programs?
SQL	MSD NMR database	- PIMS - 3D-LIMS		- (3D-LIMS) - (bioinformatics)	

For all languages:
- Metamodel
- Documentation

For all formats:
- Schemas
- I/O mappings

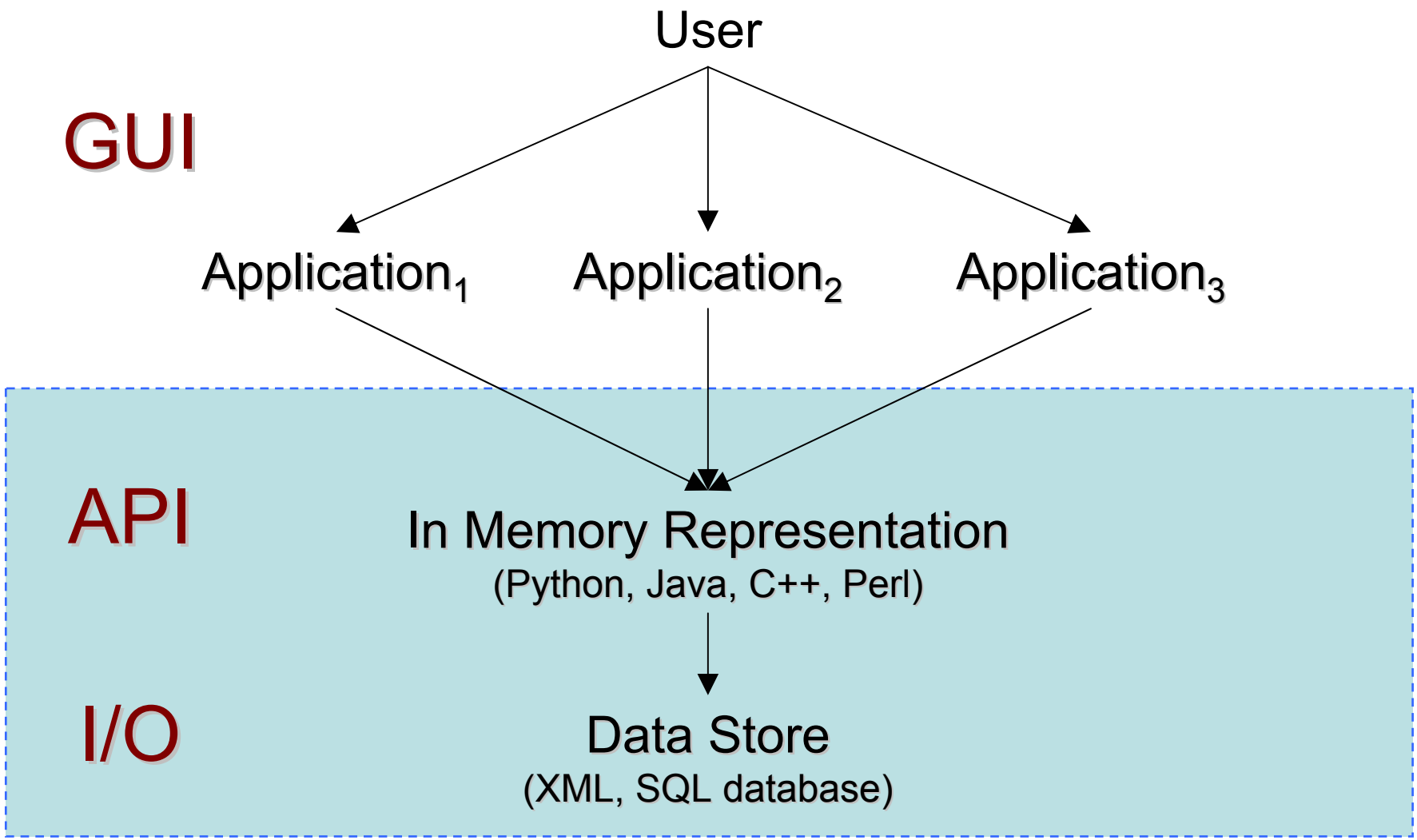
CCPN

- Introduction
- Goals and strategy
- Modeling and code generation
- The model
- APIs
- Future plans

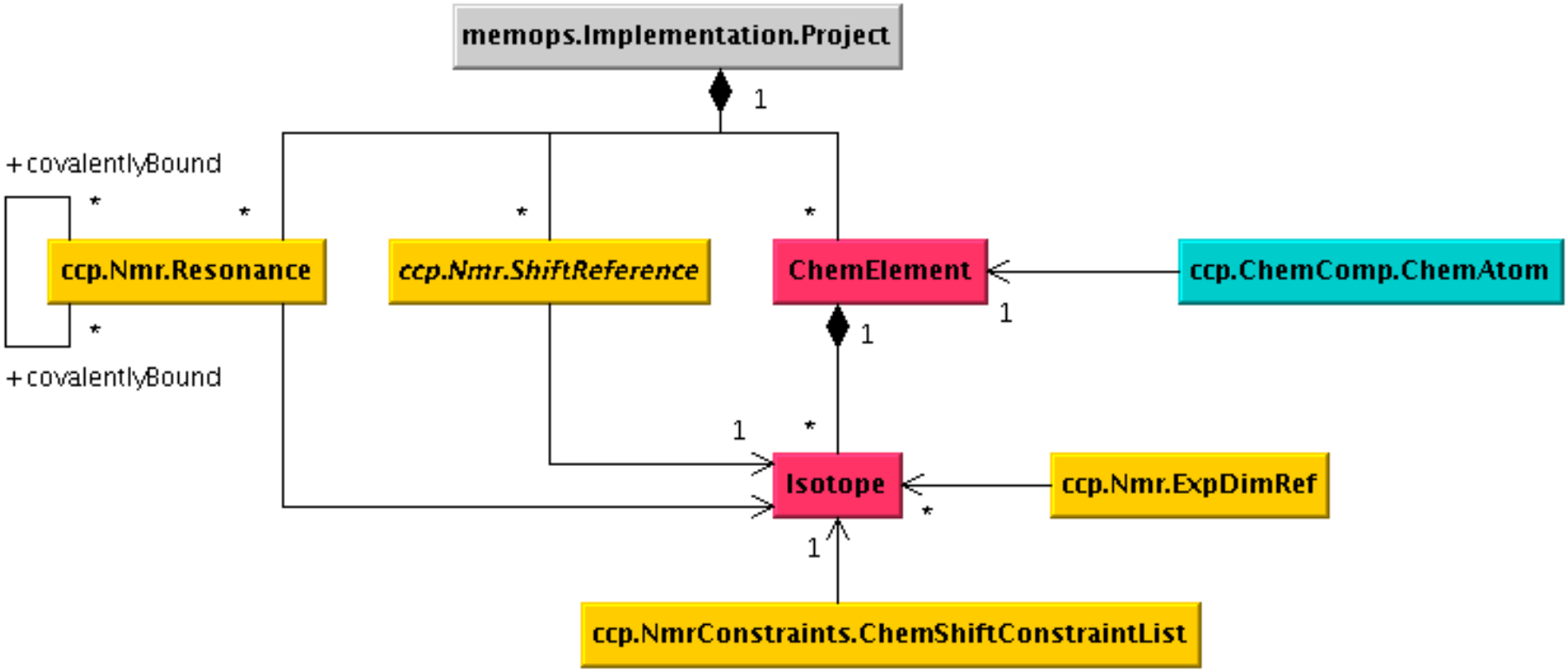
- **THE END**

CCPN

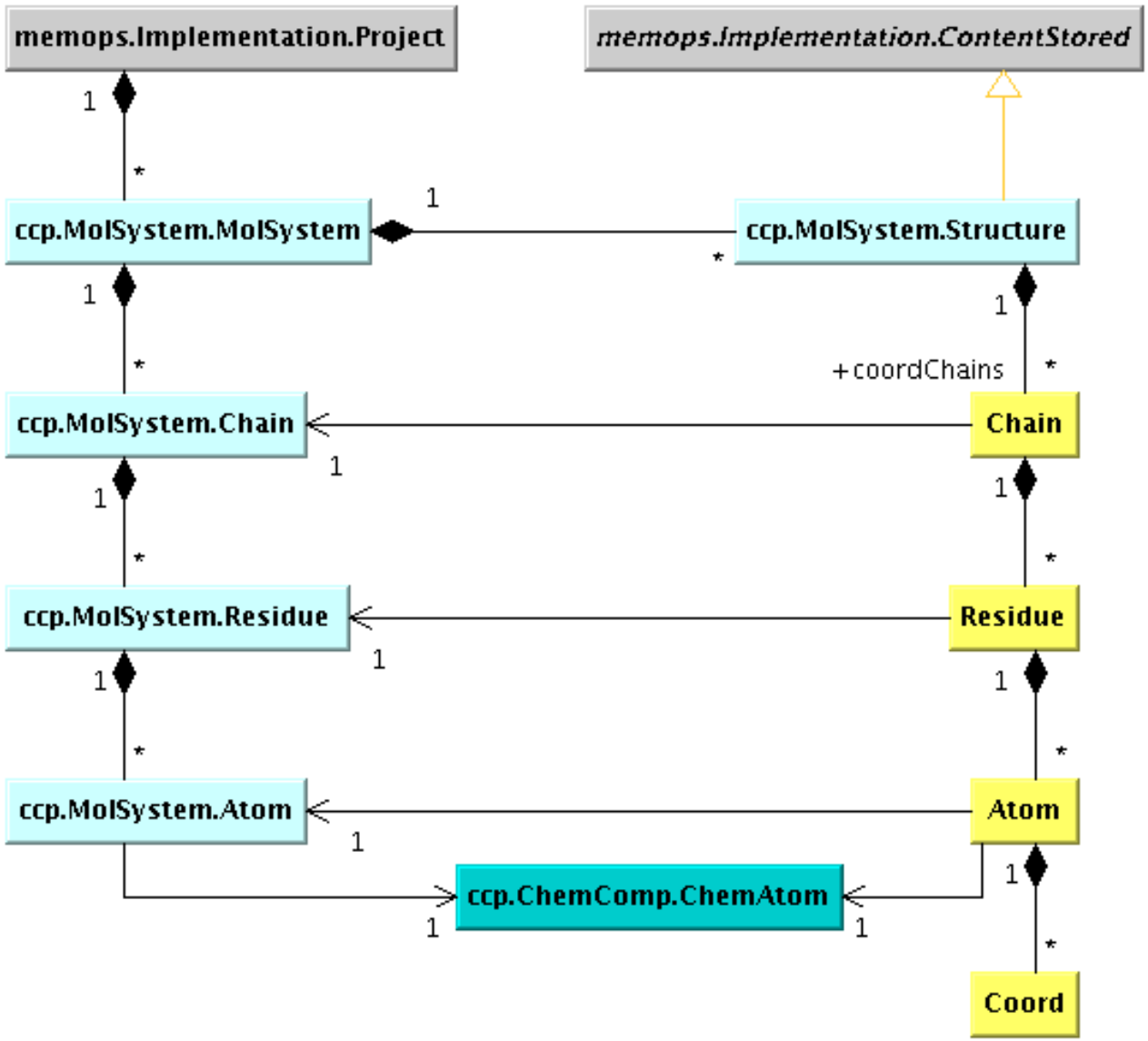
Application View



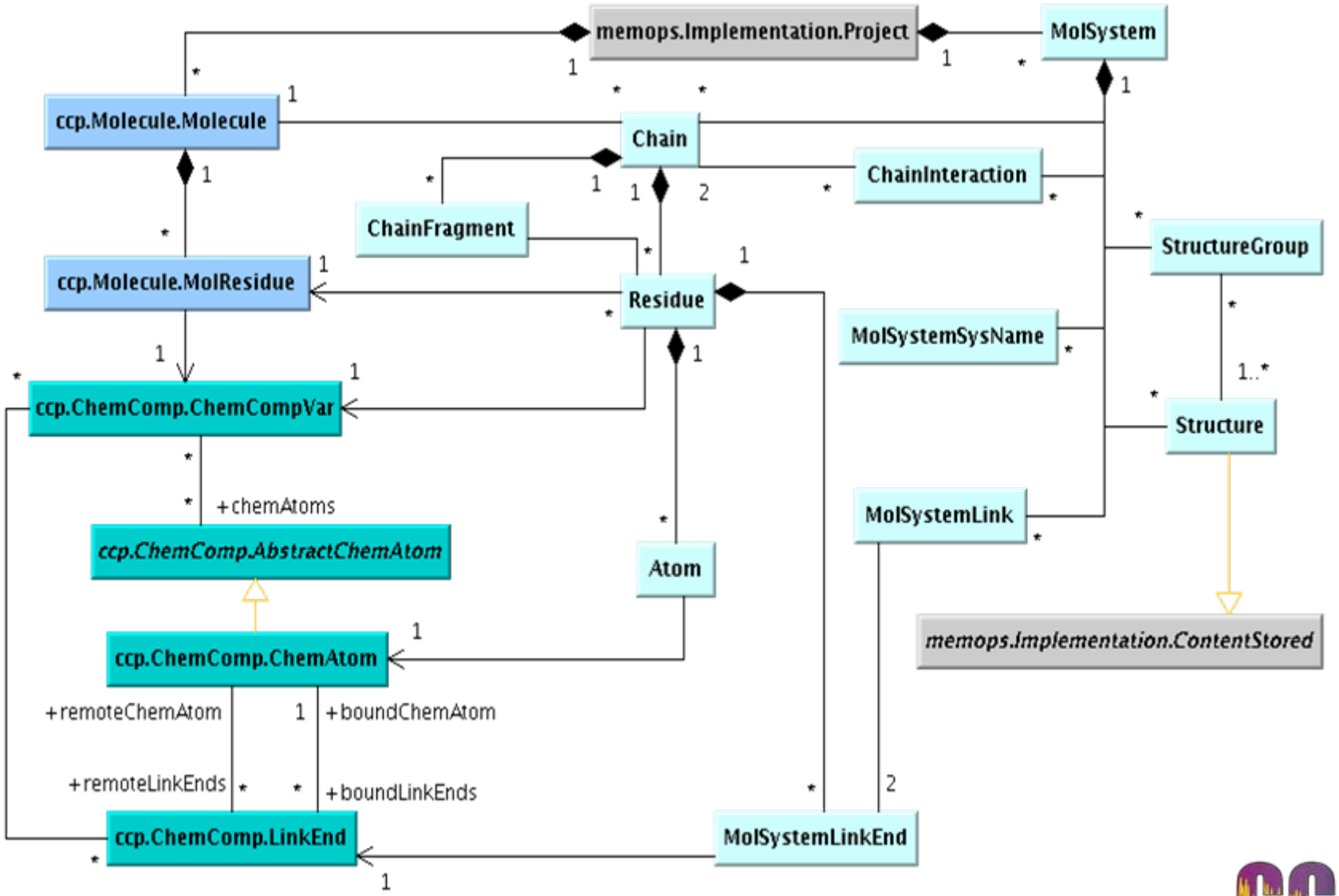
ChemElement



Coordinates



MolSystem


memops.Implementation.Project
MolSystem
ccp.Molecule.Molecule
Chain
ChainInteraction
ChainFragment
StructureGroup
ccp.Molecule.MolResidue
Residue
MolSystemSysName
ccp.ChemComp.ChemCompVar
Structure
+chemAtoms
MolSystemLink
ccp.ChemComp.AbstractChemAtom
Atom
memops.Implementation.ContentStored
ccp.ChemComp.ChemAtom
+remoteChemAtom
+boundChemAtom
+remoteLinkEnds
+boundLinkEnds
ccp.ChemComp.LinkEnd
MolSystemLinkEnd

API - Operations provided

- 'get' and 'set' *(Attributes and links)*
- 'add' and 'remove' *(Multiple attributes and links)*
- 'findFirst' and 'findAll' *(Multiple links)*
  - Simple filtering (attribute == value)

- create and 'new' *(Objects)*
  - Normal and 'factory function' object creation
- delete *(Objects)*
  - 'Delete' function. Cascades to objects that would otherwise be rendered invalid.

New Core API technology, cont.

- Remodelling of implementation details
  - Storage pointers
  - Collection types (sets, ordered sets, lists)
  - Root objects
- Complex data types
  - e.g. rotation matrix
- Longer term: Client/Server architecture
  - For PIMS and 3D-LIMS