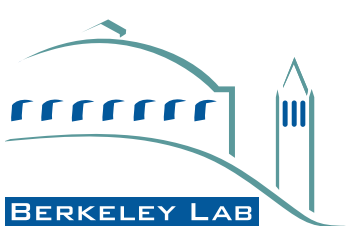
Schema Migration for BaBar's Federations

Simon Patton
and Igor Gaponenko

Lawrence Berkeley National Laboratory



Overview

- **Schema**
- **Objectivity's Evolution**
- **BaBar Criteria for Migration**
- **Implementations**

Objectivity Schema

- **Schema defined "shape" of classes in Federation**
- **Two types of classes in Schema**
  - "Persistent" class, subclasses ooObj
  - "Embeddable" class, standard C++ class
    - (but no pointers)

Simple Example

```
class LabeledLink
   : public ooObj
{
      // function declarations

   private:

      // data members
      ooVString m_label ;
      ooRef( LabeledLink ) m_link ;
};
```

```
class LabeledLink
basic
1000000 1000000
1 0 3 0 3 0 0
{
  Embedded
  : public ooObj
  1001
  1 0 0 0

  Embedded
  private ooVString m_label
  5005
  2 1 0 0

  Oid
  private ooRef(LabeledLink) m_link
  1000000
  3 2 0
}
```

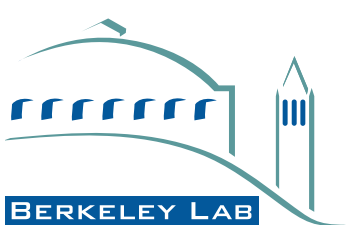
Objectivity Schema Evolution

- **Change a class and Schema has to "evolve"**
  - Objectivity manual lists 50 possible modifications that "require" schema evolution
  - e.g. Add a data member
- **Two types of evolution**
  - Conversion
  - non-Conversion
- **Will affect existing executables**

Added Data Member

```
class LabeledLink
   : public ooObj
{
      // function declarations

   private:

      // data members
      ooVString m_label ;
      ooRef( LabeledLink ) m_link ;
      d_Long m_count ;
};
```

```
class LabeledLink
basic
1000000 1000001
1 0 4 0 4 0 0
{
  Embedded
  : public ooObj
  1001
  1 0 0 0

  Embedded
  private ooVString m_label
  5005
  2 1 0 0

  Oid
  private ooRef(LabeledLink) m_link
  1000000
  3 2 0

  Basic
  private int32 m_count
  4 3 0 0
}
```



BaBar Criteria for Releases
(and thus schema migration)

- **No conversion migrations**
  - (most data is read only)
- **Any executable that works a with subset of data must continue to work *without recompilation or relinking*.**
- **Objectivity Manual States:**
  - Both conversion and non-conversion operations require you to *relink existing applications*.

BaBar Schema Migration

- **Once a class is in the Schema is may not cause schema evolution.**
  - New class required for each change of "shape".
    - Convention <classname>_xxx
- **Need to insulate clients**
  - both physicist and non-DB programmers
- **Use Transient-Persistent conversion**

Transient-Persistent Conversion (transient side)

- **Used in BaBar non-event data**
- **Transient proxies "fault" in the data upon request.**
- **"fault handles" do the work**
  - Get generic handle from Federation
  - Cast is to "real" type (using ooTypeN)
  - Executes code for that type to build transient.

Transient-Persistent Conversion (persistent side)

- **Used in BaBar EventStore**
- **Templated "converters" call persistent class to do the work**
  - Requires "standard" interface, so have pure abstract class based on transient type.
  - All actions based on this interface class, except one static "locate" class which must be class specific.
    - If class fails to "locate" instance of itself delegates task to "locate " of preceding generation of conceptual class.

Event Store Base Class

```
template< class T >
class BdbEvtObjP
   : public BdbEvtObj
{
      BdbEvtObjP( const AbsEvtObj* aTransient ,
                  BdbEvtObjLocReg& aRegistry ) ;

      virtual ~BdbEvtObjP() ;

      virtual T* transient( BdbEvtObjLocReg& aRegistry ) const = 0 ;

      virtual d_Boolean fillPointers( T* aTransient ,
                                      BdbEvtObjLocReg& aRegistry ) const = 0 ;

      virtual d_Boolean fillRefs( const T* aTransient ,
                                  BdbEvtObjLocReg& aRegistry ) = 0 ;

} ;
```

Transient-Persistent Conversion (transient wrapper)

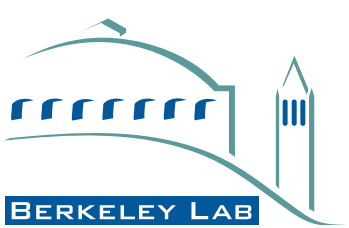
- **Used in Federation "infrastructure" code**
  - (not a true conversion)
- **Transient wrapper class do the work**
  - Initialized with handle to persistent object
    - Can be generic handle or specific base class.
  - Call to transient class forwarded to persistent class
    - Either simple forward or can execute type specific code

Transient-Persistent Conversions Comparison

- **Both used successfully in BaBar.**
- **Transient side conversion:**
  - Allows more freedom in persistent class organization
  - All conversion code in one place
- **Persistent side conversion**
  - Write conversion once
  - Classes only know about predecessor

Transient-Persistent Conversions Comparison (II)

- **Transient wrapper**
  - Allows lazy evaluation of inter-object links
  - Can be used to make execution-time links!
- **Best choice?**
  - Different horse for different course



Summary

- **To avoid relinking all code for every release can not use Objy schema evolution**
- **New class for each "shape" of a conceptual class**
- **Insulate users by Transient-Persistent conversion**
  - I find abstract base class easiest way to implement this.