

Inova

Development

The **CIMPLE** Provider Development Environment

Michael Brasher m.brasher@inovadevelopment.com

Karl Schopmeyer k.schopmeyer@inovadevelopment.com

- I. Overview
- II. Simplifying Provider Development
- III. Developing providers with **CIMPLE**
- IV. Futures

Inova

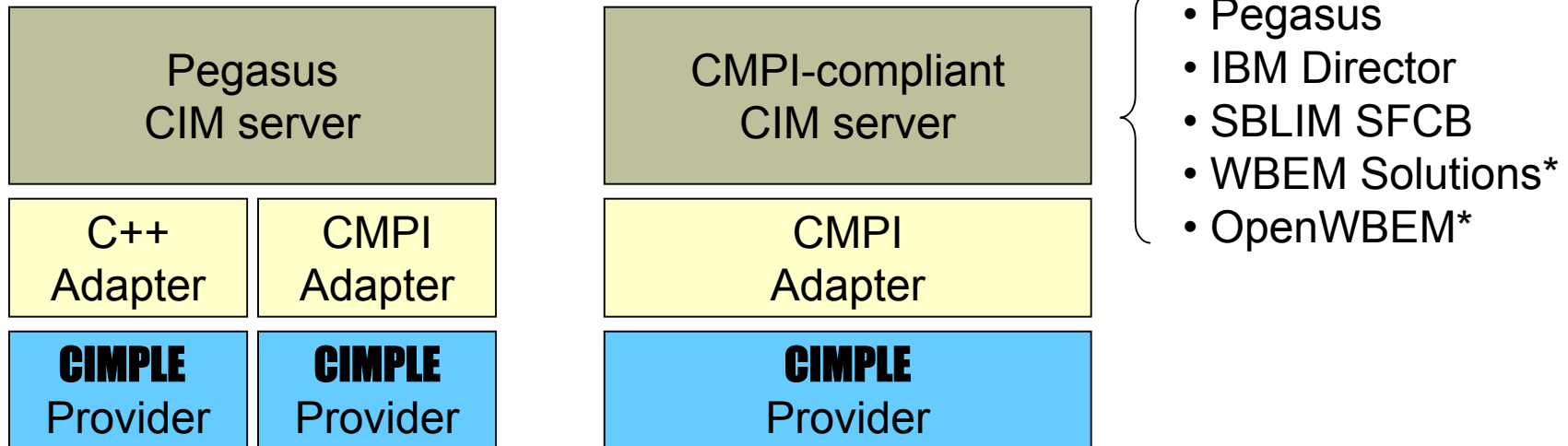
Development

Part I

Overview

What is **CIMPLE**?

- **CIMPLE** is an environment for building providers for a wide range of CIM servers.



* No interoperability testing performed against these servers yet.

- Providers **interoperate** with a wide range of CIMOMs.
- Cuts the **cost** and **effort** of developing providers substantially.
- Reduced code complexity and type safe interfaces lead to better **quality**.
- Providers are **small** and **fast**; well suited for embedded environments.

Supported Platforms

OS	Processor	Compiler
Linux	IX86-32 bit	GNU C++
Linux	IX86-64 bit	GNU C++
Hardhat Linux	IX86-32 bit	GNU C++
Darwin	PPC-32 bit	GNU C++
Darwin	IX86-32 bit	GNU C++
Windows	IX86-32 bit	MSVC 6.X
Windows	IX86-32 bit	MSVC 7.X
Windows	IX86-32 bit	MSVC 8.X
Windows	IX86-64 bit	MSVC 8.X

Major features added this year

- Portable threading library for provider.
- Pegasus adapter CMPI adapter.
- CIMOM up-calls.
- Object path processing.
- Port to PPC
- Port to Darwin
- Version stamps
- Added qualifiers to class meta data.
- Added default values to class meta data.
- Initialization of instance properties from meta data.
- Multi-class providers.
- Dynamic casting of generated classes.

Copyright (c) 2003, 2004, 2005, 2006, Michael Brasher, Karl Schopmeyer

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

- “Free” **CIMPLE** users can expect the following.
 - **CIMPLE** may be used and redistributed free of charge.
 - Anyone can access ongoing work with the public CVS code repository.
 - We answer support questions as long as doing so is only a minor effort on our part.
 - Anyone can report bugs or make enhancement requests via Bugzilla.
 - Critical bugs are fixed for the next public release. Other bugs are fixed at our discretion.
 - Enhancement requests are implemented at our discretion.

- Clients with support contracts are entitled to the following.
 - Quick turnaround on support questions.
 - Quick resolution to bugs.
 - Quick implementation of enhancements.
 - Set development priorities.
 - Drive future direction and architecture.

- **CIMPLE** e-mail addresses:
 - m.brasher@inovadevelopment.com
 - k.schopmeyer@inovadevelopment.com
 - cimplesupport@inovadevelopment.com

CIMPLE.org

[Home](#) | [Documentation](#) | [Downloads](#) | [About Us](#) | [Contact Us](#) | [Links](#) | [Bugs](#)



[Contents](#)

[CIMPLE Support](#)

[Karl Schopmeyer](#)

[Mike Brasher](#)

Contact Us



CIMPLE Support
cimplesupport@inovadevelopment.com

Karl Schopmeyer (Project Manager)
kschopmeyer@inovadevelopment.com
(972) 814-5581

Mike Brasher (Project Architect)
mbrasher@inovadevelopment.com
(512) 219-9480

- CIMPLE can be obtained from <http://cimple.org>.
- You can obtain the latest source distribution or check it out from the CVS repository.

Inova

Development

Part II

Simplifying Provider Development

Class generation

- Using generated classes substantially reduces code complexity by eliminating code needed to:
 - “Lookup” a property from an instance.
 - Check whether a property exists.
 - Check for type mismatches.



Example 1

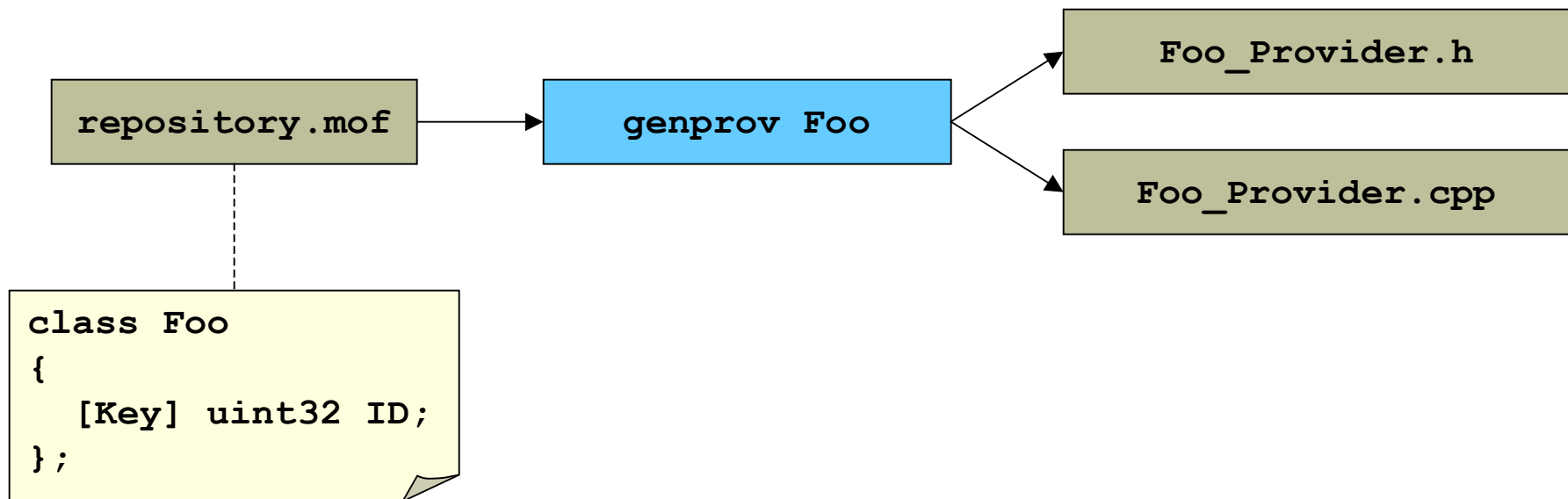
```
// CIMPLE:  
uint32 value = foo->ID.value;  
bool null = foo->ID.null;
```

```
// CMPI:  
CMPIData data;  
CMPIStatus status;  
CMPIUint32 value;  
int null = 0;  
  
data = CMGetProperty(foo, "ID", &status);  
  
if (status.rc != CMPI_RC_OK)  
{  
    // No such property! Handle error!  
}  
  
if (data.type != CMPI_uint32)  
{  
    // Type mismatch! Handle error!  
}  
  
if (data.state & CMPI_nullValue)  
    null = 1;  
  
value = data.value.uint32;
```

```
// CIMPLE:  
uint32 value = foo->ID.value;  
bool null = foo->ID.null;
```

```
// Pegasus:  
Uint32 value;  
Boolean null;  
  
Uint32 pos = foo.getProperty("ID");  
  
if (pos == PEG_NOT_FOUND)  
{  
    // No such property! Handle error!  
}  
  
try  
{  
    CIMProperty cp = foo.getProperty(pos);  
    CIMValue cv = cp.getValue();  
    null = cv.isNull();  
    cv.get(value);  
}  
catch (Exception& e)  
{  
    // Type mismatch! Handle error!  
}
```

- **CIMPLE** generates the provider skeletons automatically.
- The provider developer implements one or two skeletons in a typical case.



Generated Provider Class

```
class Foo_Provider
{
public:
    Foo_Provider();

    ~Foo_Provider();

    Load_Status load();

    Unload_Status unload();

    Get_Instance_Status get_instance(
        const Foo* model,
        Foo*& instance);

    Enum_Instances_Status enum_instances(
        const Foo* model,
        Enum_Instances_Handler<Foo>* handler);

    Create_Instance_Status create_instance(const Foo* instance);

    Delete_Instance_Status delete_instance(const Foo* instance);

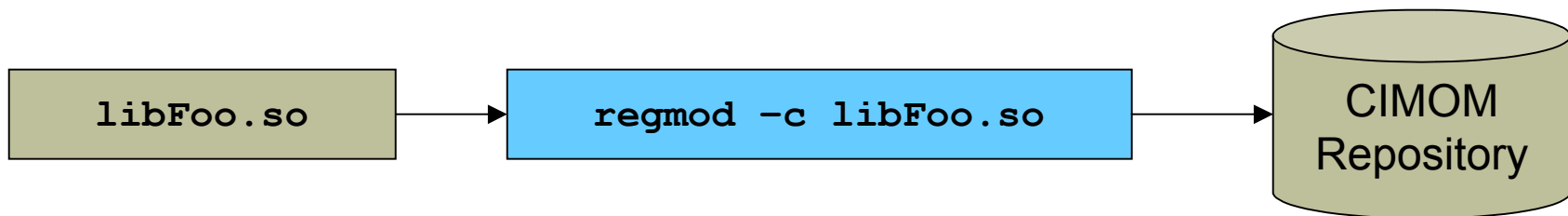
    Modify_Instance_Status modify_instance(const Foo* instance);
};
```

Operation Reduction

Operation	Implementation	Notes
Get-instance	Automated	Automated via Enumerate-Instance if not implemented.
Get-instance-names	Deprecated	Implemented via Get-instance
Enum-instances	Required	Required by instance providers.
Enum-instance-names	Deprecated	Implemented via Enum-instance
Create-instance	Optional	Provider dependent
Modify-instance	Optional	Provider dependent
Delete-instance	Optional	Provider dependent
Associators	Deprecated	Implemented via Associator-names
Associator-names	Automated	Automated via Enumerate-instances not implemented.
References	Automated	Automated via Enumerate-instances not implemented.
Reference-names	Deprecated	Implemented via References
Invoke-Method	Required	Required by method providers.
Enable-indication	Required	Required by indication providers.
Disable-indication	Required	Required by indication providers.

- **CIMPLE** automates the following operations via enumerate-instances, if the provider developer opts not to implement them.
 - Get-instance
 - Associator-names
 - References

- Regmod tool automates provider registration and class creation.



1. Regmod only works with Pegasus today.

Elimination of Object Paths

- In **CIMPLE**, object paths are represented with ordinary instances. Only the key fields are used. This eliminates the confusing dichotomy between an object path and an instance. For example.

```
class MyClass
{
  [Key] uint32 Key1;
  [Key] string Key2;
  uint32 Prop3;
  string Prop4;
  boolean Prop5;
};
```

```
// MyClass.Key1=99,Key2="Hello"
MyClass* keys = MyClass::create();
keys->Key1.value = 99;
keys->Key2.value = "Hello";
```

Inova

Development

Part III

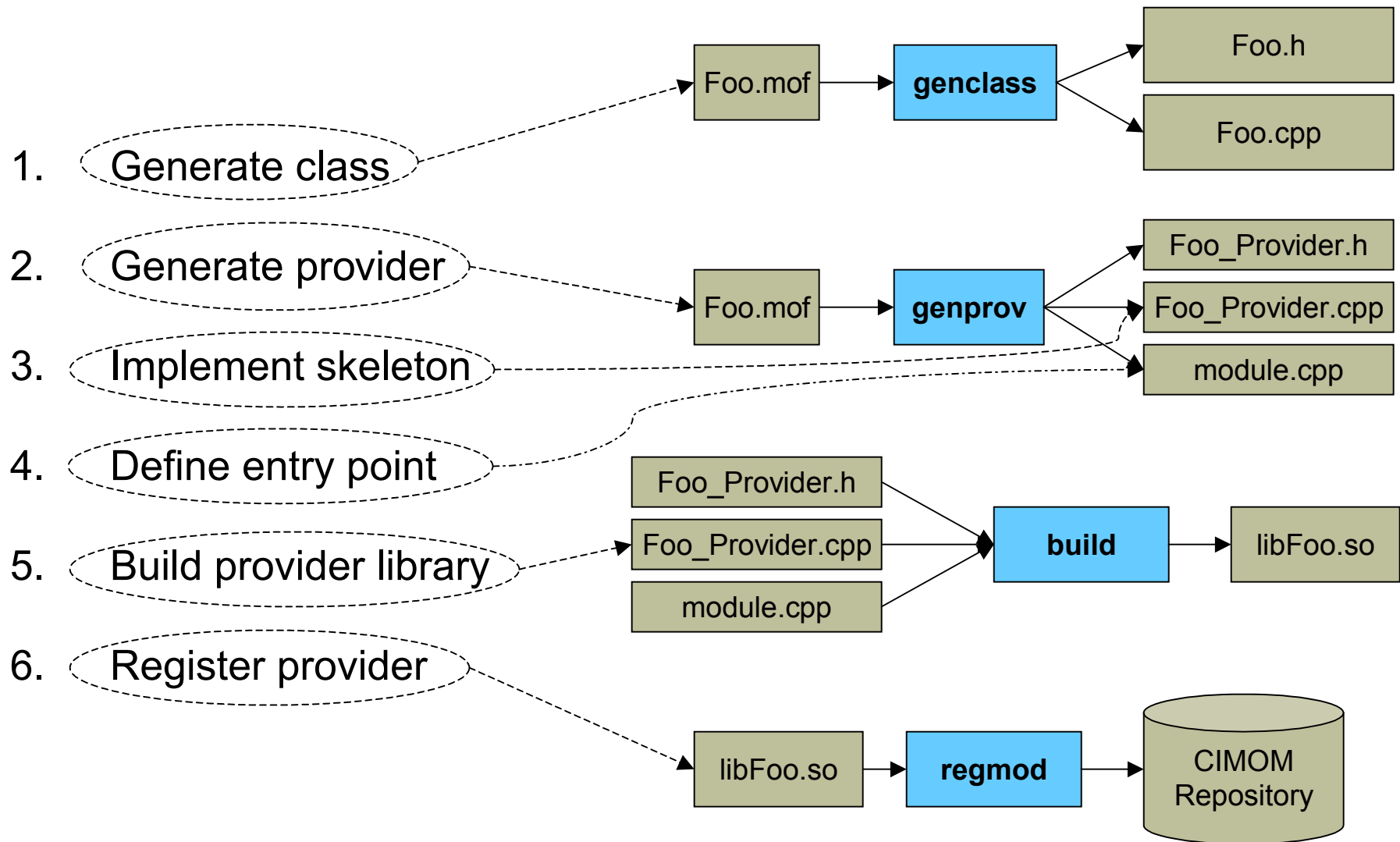
Developing Providers with **CIMPLE**

- **Instance Provider**
- **Association Provider**
- **Indication Provider**
- **Method Provider** – not a distinct provider type in **CIMPLE**, since any of the above provider types may implement methods.

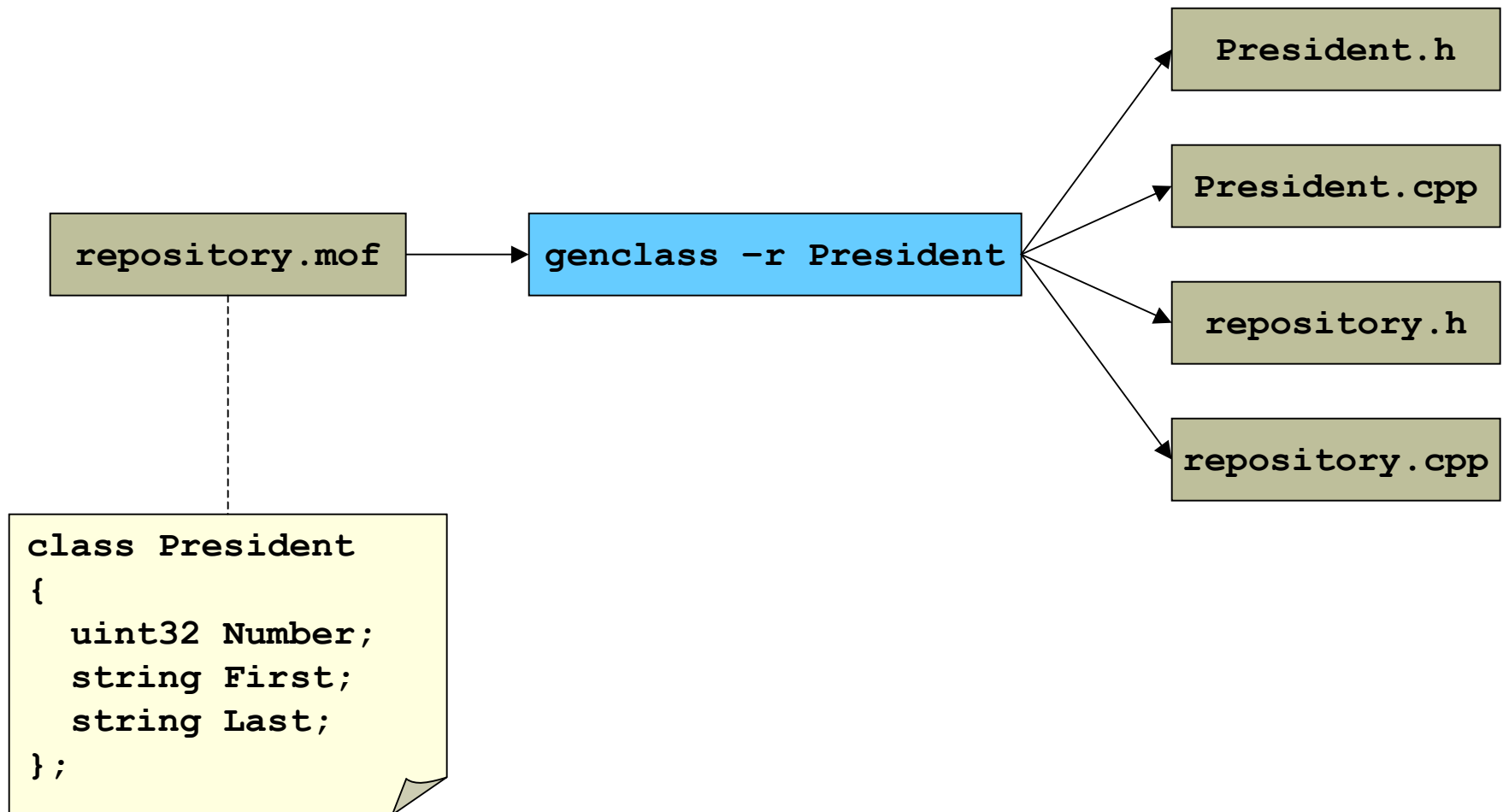
Provider Operations

Operation	Instance Provider	Association Provider	Indication Provider
load()	✓	✓	✓
unload()	✓	✓	✓
get_instance()	✓	✓	✓
enum_instances()	✓	✓	✓
create_instance()	✓	✓	✓
modify_instance()	✓	✓	✓
delete_instance()	✓	✓	✓
enum_associators()		✓	
enum_reference_names()		✓	
enable_indications()			✓
disable_indications()			✓
extrinsic methods	✓	✓	✓

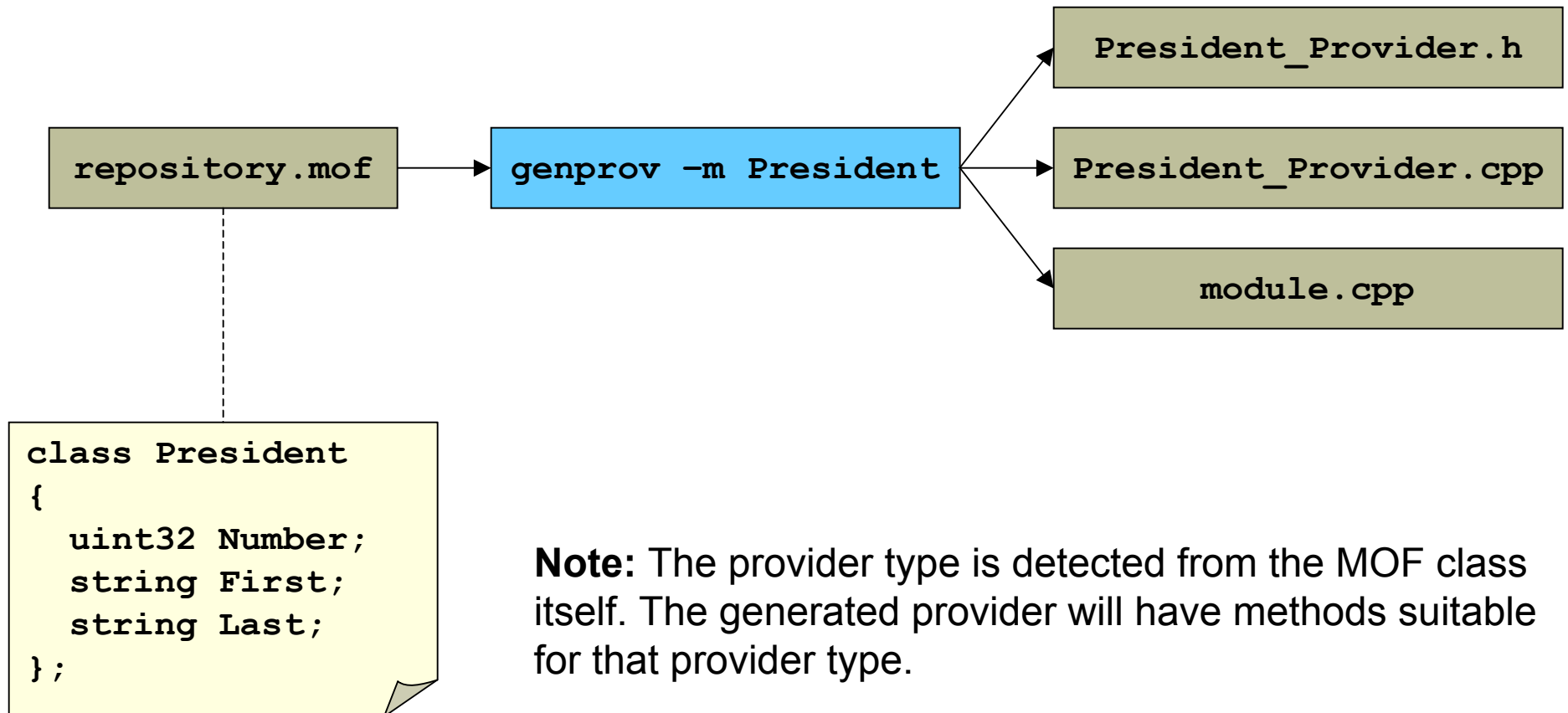
Provider development steps



1. Generate class



2. Generate provider



3. Implement skeleton

President_Provider.cpp

```
Enum_Instances_Status President_Provider::enum_instances(  
    const President* model,  
    Enum_Instances_Handler<President>* handler)  
{  
    President* p1 = President::create();  
    p1->Number.value = 1;  
    p1->First.value = "George";  
    p1->Last.value = "Washington";  
    handler->handle(p1);  
  
    President* p2 = President::create();  
    p2->Number.value = 2;  
    p2->First.value = "John";  
    p2->Last.value = "Adams";  
    handler->handle(p2);  
  
    return ENUM_INSTANCES_OK;  
}
```

Developer
writes
this

4. Define entry point

module.cpp

Generated

```
#include "President_Provider.h"
```

```
using namespace cimple;
```

```
CIMPLE_MODULE(President_Module);
```

```
CIMPLE_PROVIDER(President_Provider);
```

Developer
writes
this

```
// Define CMPI entry point.
```

```
CIMPLE_CMPI_INSTANCE_PROVIDER(President_Provider);
```

```
// Define Pegasus provider entry point.
```

```
// CIMPLE_PEGASUS_PROVIDER_ENTRY_POINT;
```

1. Could also define entry point for Pegasus C++ provider interface

5. Build provider

Makefile

```
TOP = $(CIMPLE_ROOT)
include $(TOP)/mak/config.mak

SHARED_LIBRARY = President

SOURCES = repository.cpp President.cpp module.cpp \
  President_Provider.cpp

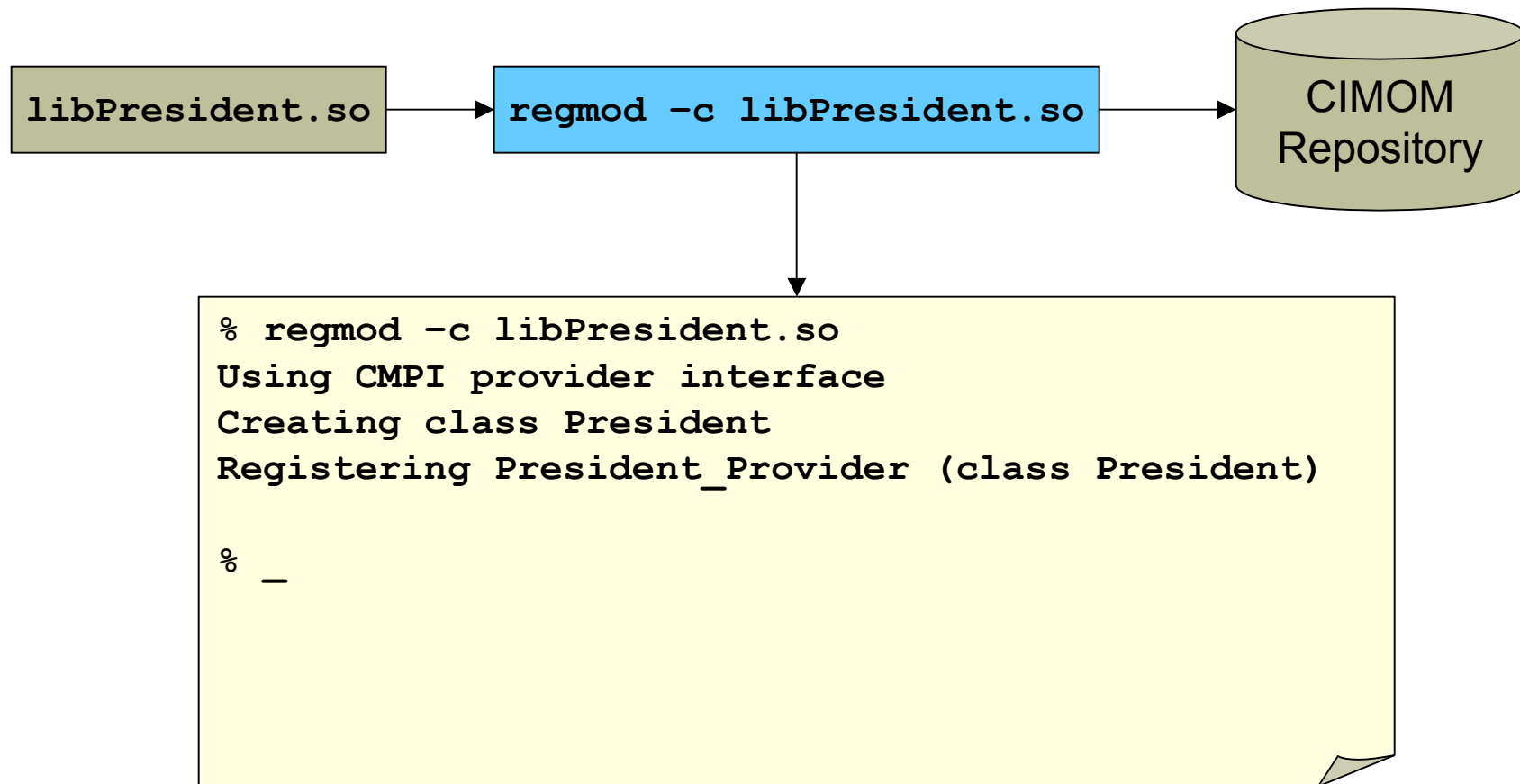
LIBRARIES += cimple cimplecmpiadap

# Link with the CMPI adapter.
LIBRARIES += cimplecmpiadap

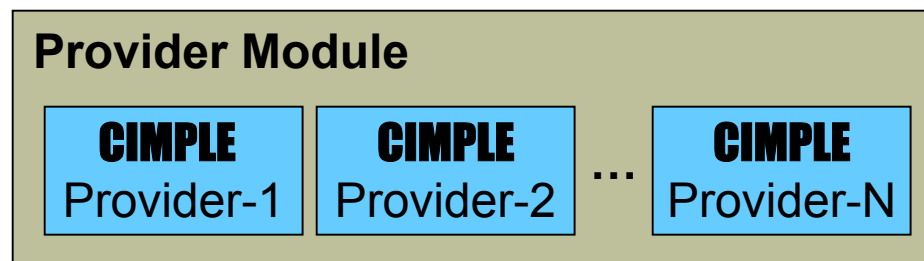
# Link with the Pegasus adapter.
# LIBRARIES += cimplepegadap

include $(TOP)/mak/rules.mak
```

6. Register provider



- CIMPLE allows several providers to be packages together in a **provider module**.
- Modules are contained in a single shared library.
- The regmod utility discovers and registers all providers in a provider module.



- Genprov generates method skeletons for each method encountered in the MOF class.



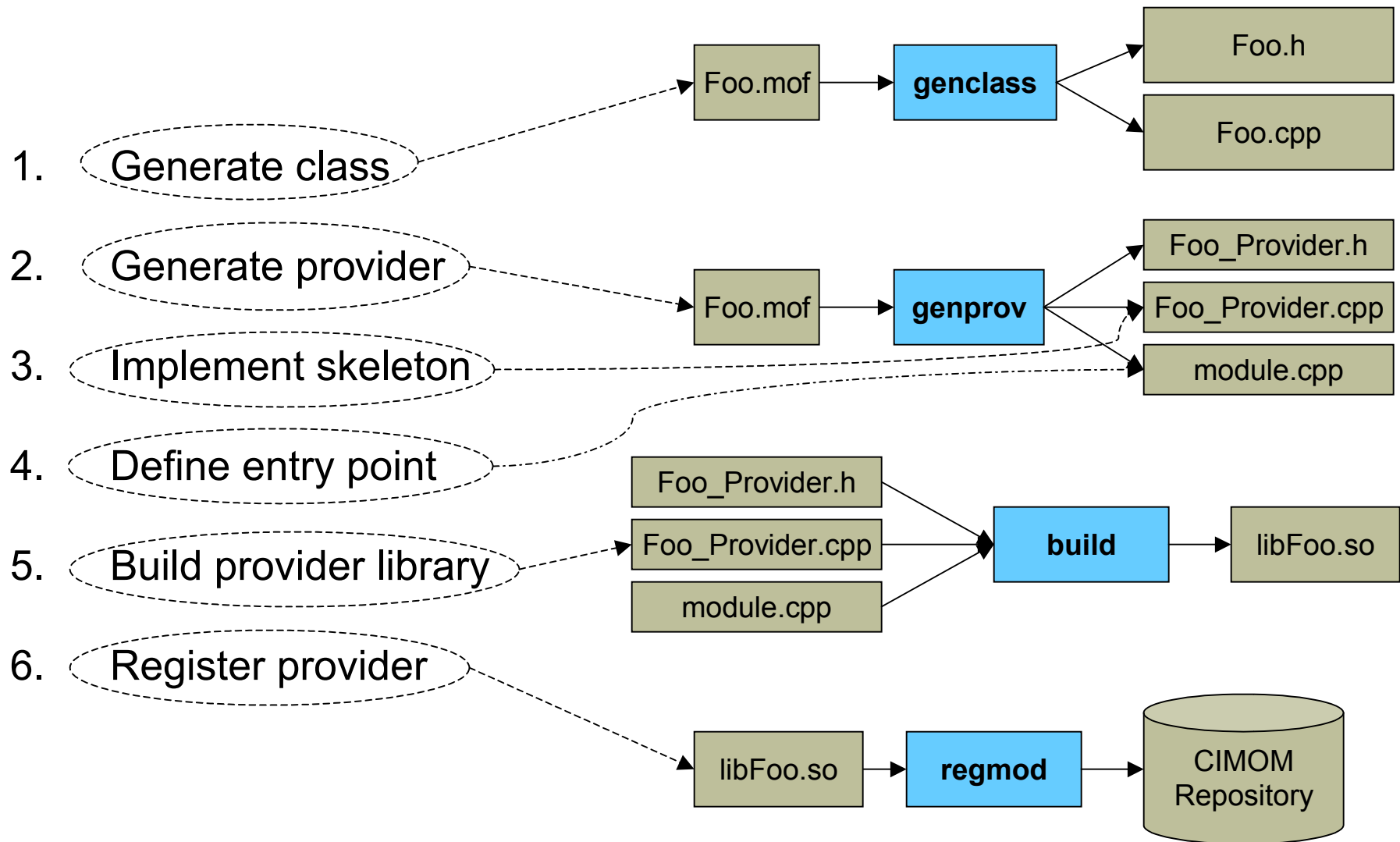
```
class Foo
{
  [Key] uint32 key;
  uint32 foo(
    [In] string p1,
    [In(false), Out] string p2);
};
```

```
Invoke_Method_Status Foo_Provider::foo(
  const Foo* self,
  const Property<String>& p1,
  Property<String>& p2,
  Property<uint32>& return_value)
{
  return INVOKE_METHOD_UNSUPPORTED;
}
```

Developer
writes
this

```
Invoke_Method_Status Foo_Provider::foo(  
    const Foo* self,  
    const Property<String>& p1,  
    Property<String>& p2,  
    Property<uint32>& return_value)  
{  
    // "p2 = p1" will work too.  
    p2.value = p1.value;  
    p2.null = p1.null;  
  
    return_value.value = 0;  
    return_value.null = false;  
  
    return INVOKE_METHOD_OK;  
}
```

Provider development steps (again)



Inova

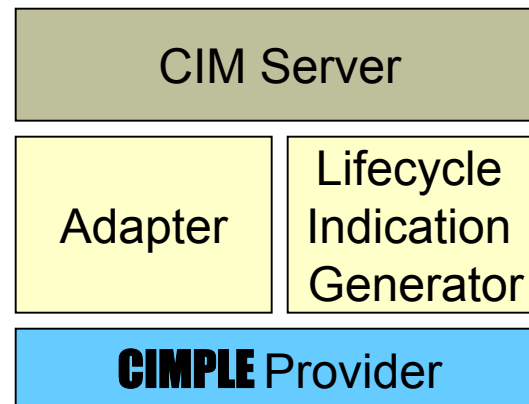
Development

Part IV

CIMPLE Futures

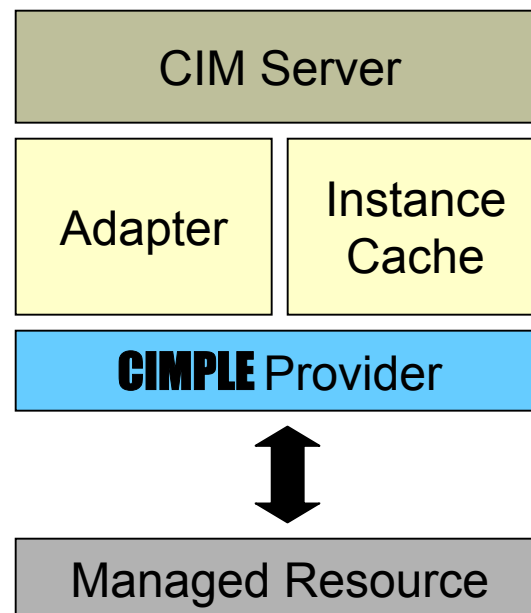
- Continue enhancing and improving **CIMPLE**.
- Develop **CIMPLE**-related products.
 - **CIMBION** – client counterpart to **CIMPLE** (i.e., an easy to use CIM client based on “first class objects” concept).
 - **CIMEngine** – a lightweight CIM server that loads **CIMPLE** providers.
 - Profile-level provider development/deployment.

- Generate lifecycle indications automatically for any provider using two modes.
 - **Transparent** – provider developer does no work to support lifecycle indications.
 - **Assisted** – provider developer inserts minimal function calls to generate lifecycle indications.



CIM Instance Caching

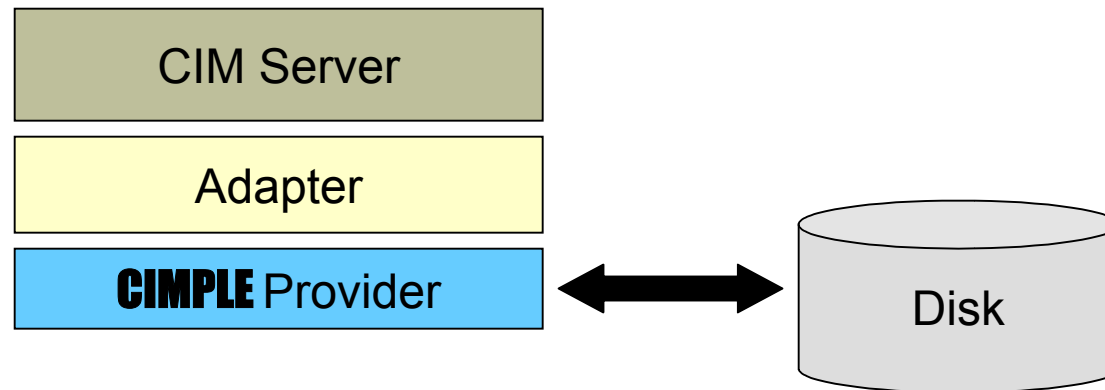
- Develop instance cache to reduce resource access overhead. No extra provider code needed to support caching.



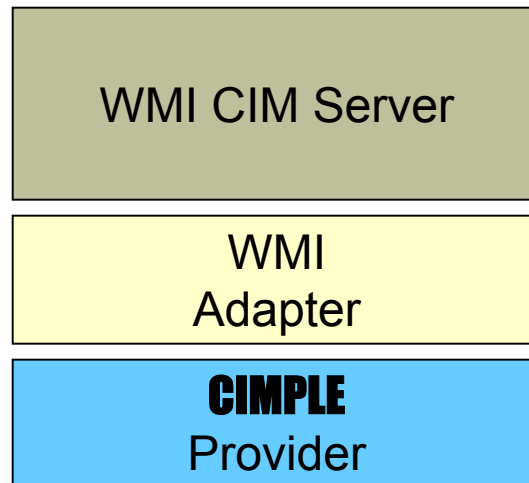
- Develop a query provider to handle query requests from the CIM server (CQL and WQL).

- Develop tracing and logging facilities to assist in developing and debugging providers.

- Develop mechanisms enabling providers to persist information (e.g., configuration information and instances).



- Develop an adapter enabling **CIMPLE** providers to work transparently with WMI.



Inova

Development

Questions

