



**December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA**

# **CIMPLE and BREVITY**

Michael Brasher [m.brasher@inovadevelopment.com](mailto:m.brasher@inovadevelopment.com)

Karl Schopmeyer [k.schopmeyer@inovadevelopment.com](mailto:k.schopmeyer@inovadevelopment.com)





# Agenda

- I. Introduction
- II. CIMPLE + demo
- III. BREVITY + demo
- IV. Work in Progress
- V. Questions

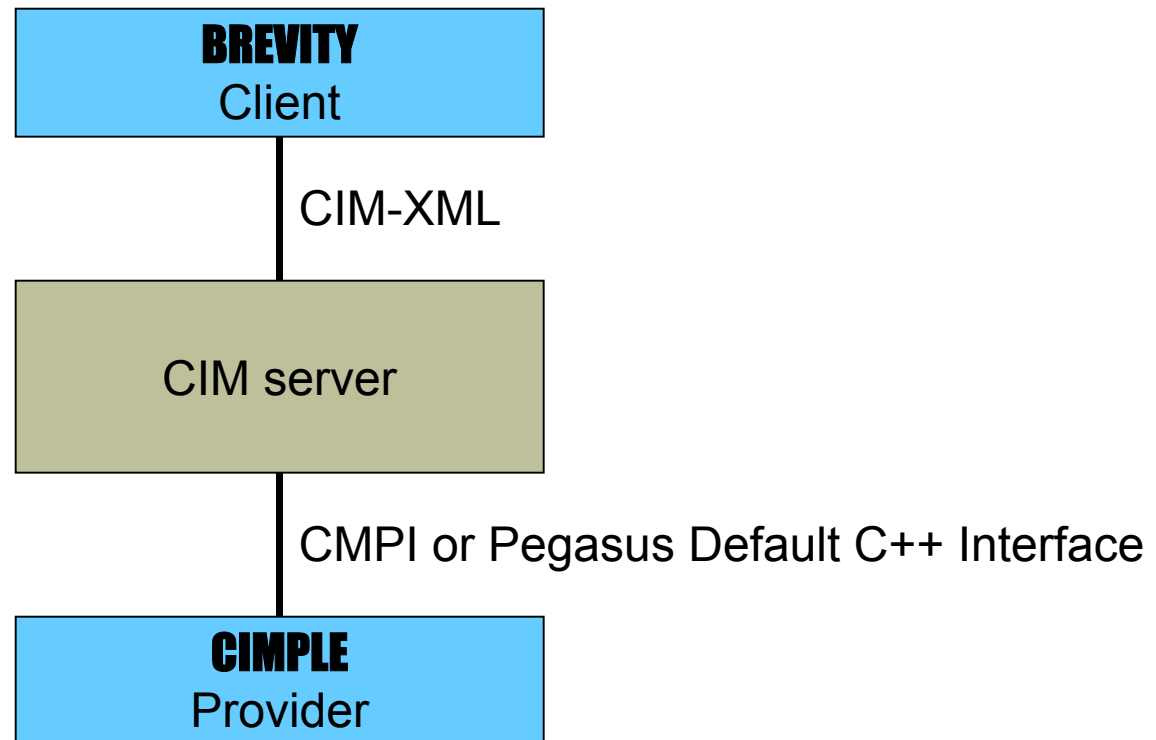


**December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA**

# **Introduction**

# What are CIMPLE and BREVITY?

- **CIMPLE** is a tool for building CIM providers.
- **BREVITY** is a tool for building CIM clients.



# Concrete CIM Elements

- The **CIMPLE** and **BREVITY** interfaces employ **concrete CIM elements** whereas conventional interfaces use **abstract CIM elements**.

```
// Abstract CIM elements in Pegasus  
CIMInstance ci("Person");  
ci.addProperty(CIMProperty(  
    "Id", Uint32(1000)));  
ci.addProperty(CIMProperty(  
    "First", String("Homer")));  
ci.addProperty(CIMProperty(  
    "Last", String("Simpson")));
```

```
// Concrete CIM elements in BREVITY:  
Person_Hnd ph;  
ph.Id_value(1000);  
ph.First_value("Homer");  
ph.Last_value("Simpson");
```

# Goals

- **CIMPLE** and **BREVITY** share the following goals:
  - Reduce code complexity.
  - Decrease development effort.
  - Reduce coding errors.
  - Improve reliability.
  - Promote interoperability.



# MIT License

- CIMPLE and BREVITY released under MIT open-source license.

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.



# MIT License Highlights

- Minimal restrictions on use.
- Free to redistribute but license header must be retained in all sources.
- No warranty license.

# Supported Platforms

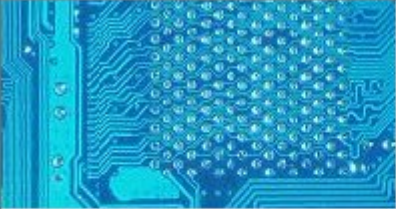
OS	Processor	Compiler
Linux	IX86-32 bit	GNU C++
Linux	IX86-64 bit	GNU C++
Linux	PPC-32 bit	GNU C++
Linux	PPC-64 bit	GNU C++
Windows	IX86-32 bit	MSVC++
Windows	IX86-64 bit	MSVC++
Darwin	IX86-32 bit	GNU C++
Solaris	SPARC	GNU C++
VxWorks 6.X	XScale	GNU C++
<b>VxWorks 5.X</b>	<b>Pentium</b>	<b>GNU C++</b>



<http://cimple.org>


**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](mailto:cimplesupport@inovadevelopment.com)

**Karl Schopmeyer (Project Manager)**  
[kschopmeyer@inovadevelopment.com](mailto:kschopmeyer@inovadevelopment.com)  
(972) 814-5581

**Mike Brasher (Project Architect)**  
[mbrasher@inovadevelopment.com](mailto:mbrasher@inovadevelopment.com)  
(512) 219-9480



# Free Support

- Answer questions via e-mail sent to [cimplesupport@inovadevelopment.com](mailto:cimplesupport@inovadevelopment.com).
- Accept patches subject to review and approval.
- Accept bug reports (open Bugzilla database).
- Fix bugs based on our own priorities and objectives.



# Contracted Support

- Paid support is available through Inova Development. Paid support includes:
  - Quick turnaround on support questions.
  - Quick resolution to bugs.
  - Implementation of custom enhancements.
  - Special releases to address client needs.
  - “Emergency” support for tight delivery schedules.



# Obtaining CIMPLE and BREVITY

- CIMPLE is available:
  - Download source distributions and installable binaries from <http://cimple.org>.
  - Checkout from the public CVS repository.
  - As part of the WBEM Solutions SDK.
  - Experimental OmniWBEM distribution (includes OpenPegasus, CIMPLE, and BREVITY).

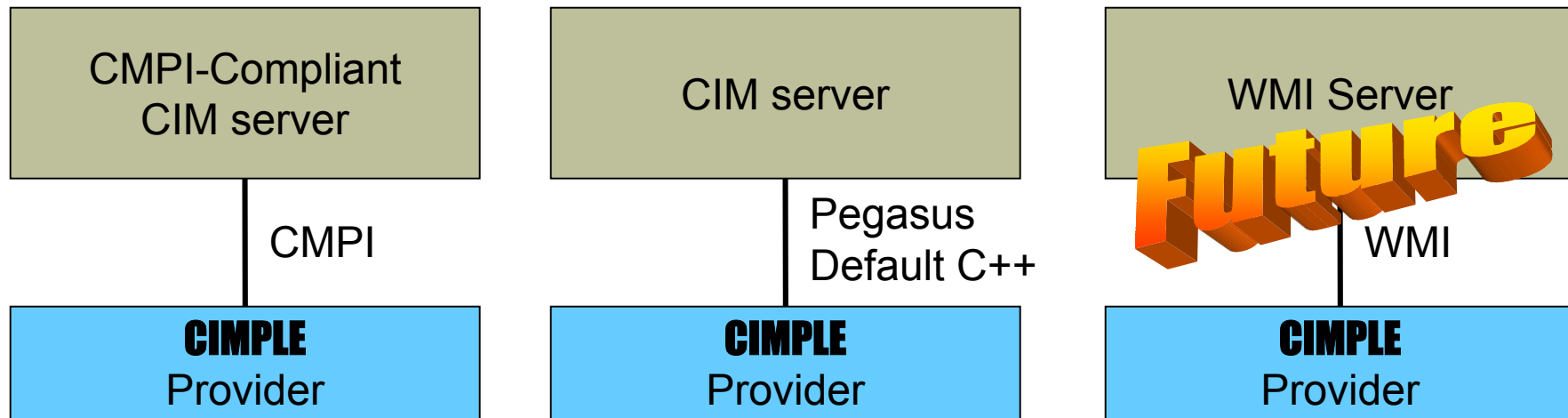


**December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA**

# **CIMPLE**

# What is CIMPLE?

- **CIMPLE** is an open-source tool for building CIM providers for multiple provider interfaces (and multiple CIM servers).
- **CIMPLE** providers can be reconfigured to support multiple provider interfaces (without source code changes).





# CIMPLE CMPI Providers

- CMPI Providers created with CIMPLE are true CMPI providers.
  - They implement the CMPI provider interface.
  - They define the proper CMPI entry points.
  - They can be loaded by CMPI-capable CIM servers.



# CIMPLE Pegasus C++ Providers

- Pegasus Default C++ Providers created with CIMPLE are true Pegasus providers.



# CIMPLE Advantages

- Support for multiple provider interfaces
  - CMPI
  - Pegasus C++ provider interface
  - WMI (future)
  - ...
- Support for multiple CIM servers.
- Less development effort and cost.
- Reduced code complexity.
- Fewer development errors.
- Smaller and faster providers (embedded systems).



# CIMPLE Provider Types

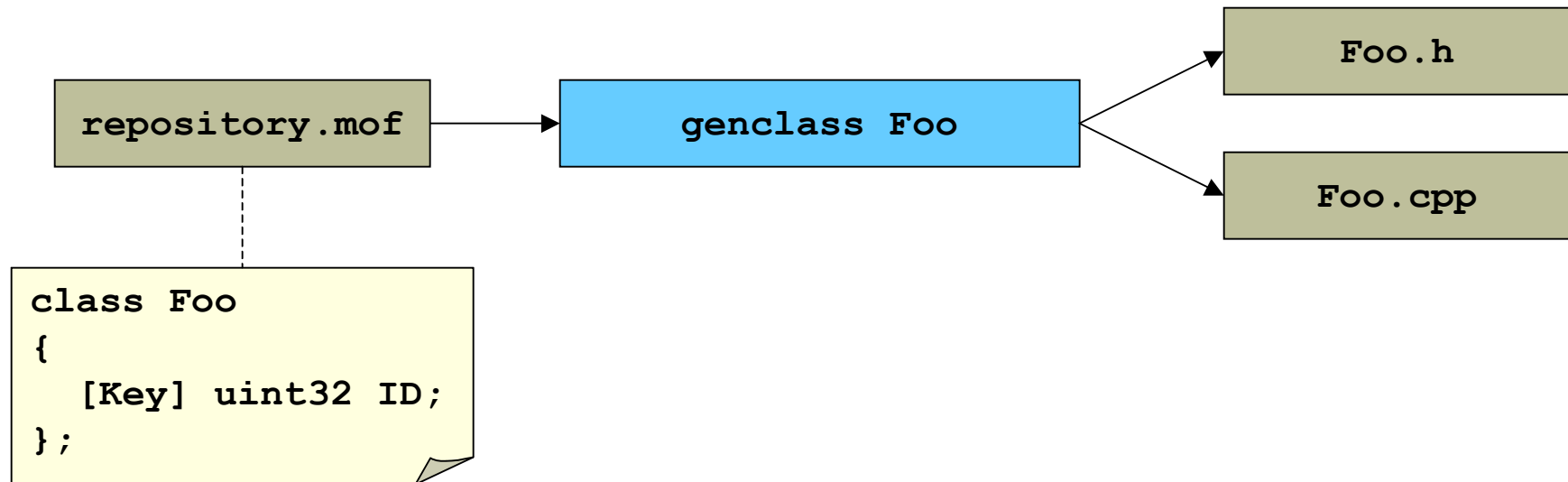
- Instance Providers
- Association Providers
- Method Providers
- Indication Providers



# Major Features

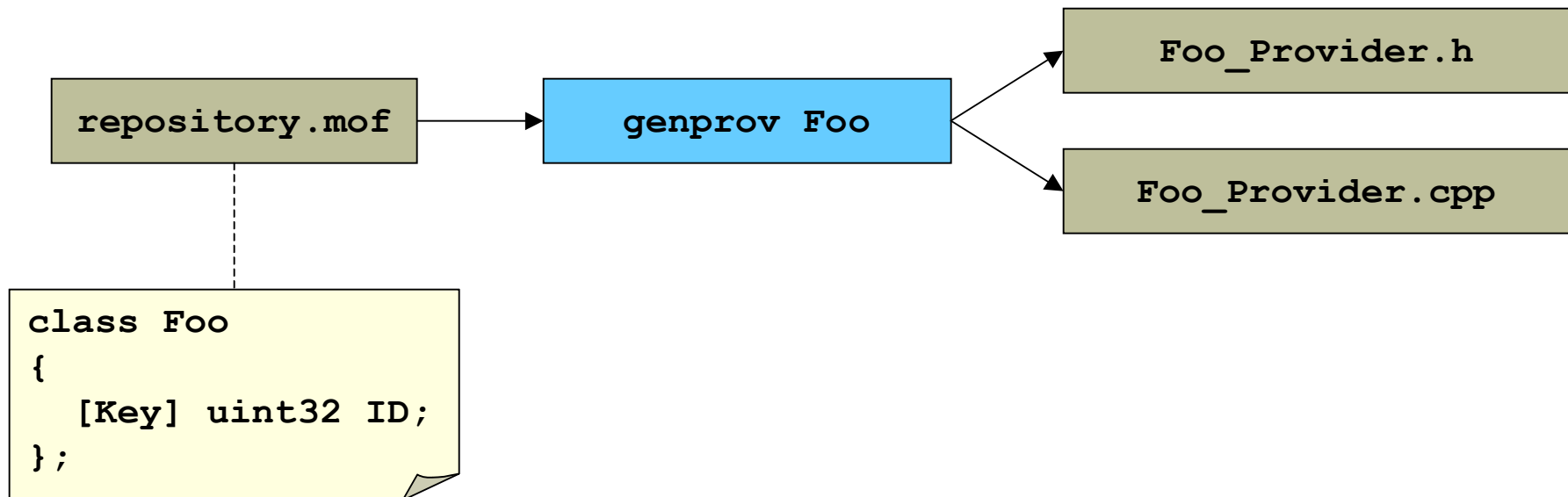
- C++ Class generation (from MOF).
- Provider skeleton generation.
- Extrinsic method stub generation.
- Module generation.
- Provider module makefile generation.
- Automated provider registration.
- Elimination of object paths.
- Operation automation.
- Module packaging.

# Class Generation

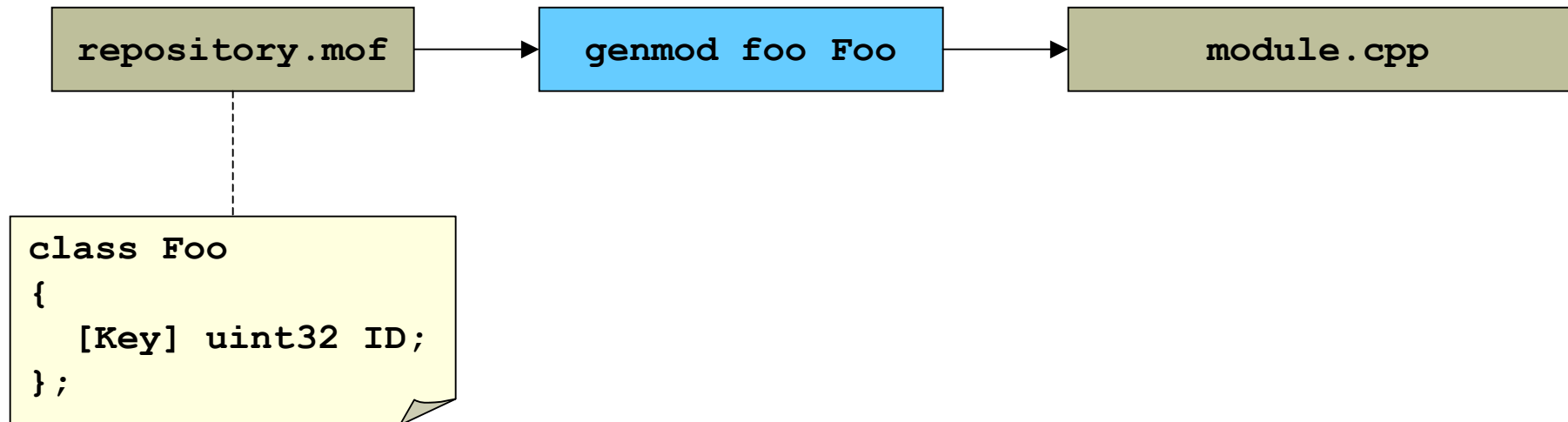


# Provider Generation

- **CIMPLE** generates the provider skeletons automatically for provider operations and extrinsic methods.

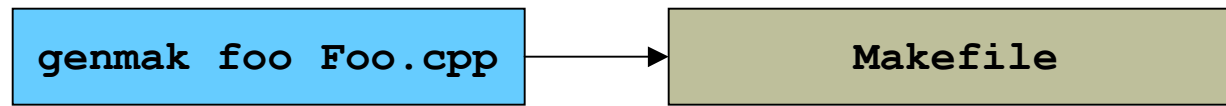


# Module Generation



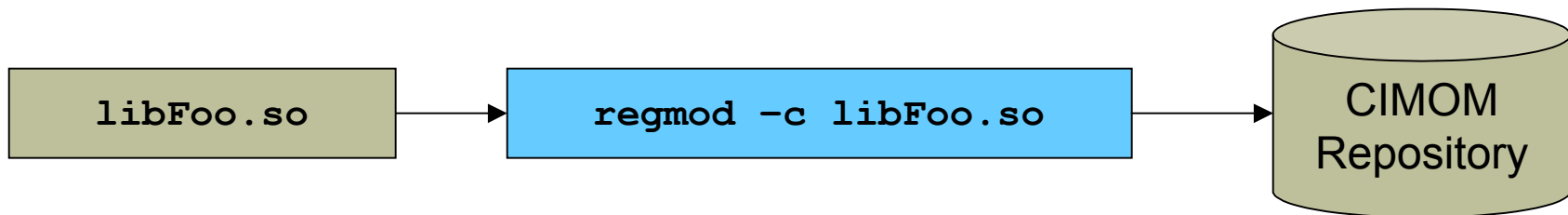


# Provider Module Makefile Generation



# Provider Module Registration

- Regmod tool automates provider registration and class creation.



1. Regmod only works with Pegasus today.

# Object Path Elimination

- 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";
```

# Operation Elimination

- **CIMPLE** eliminates the need for several operations.
  - Get-instance-names
  - Enumerate-instance-names
  - Associator-names
  - Reference-names
  - Invoke-method

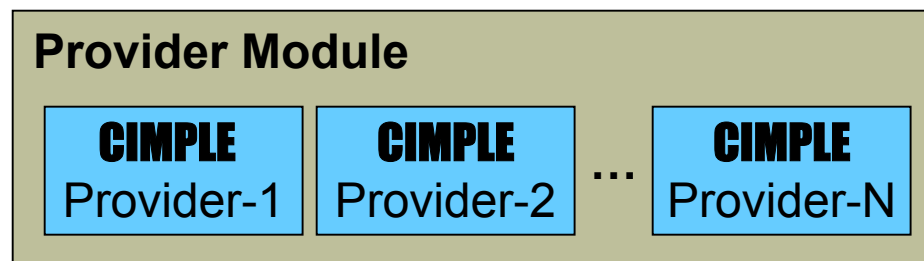


# Operation Automation

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

# Provider Modules

- 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.





# New Major Features

- New genproj tool (generates project).
- New genmak tool (generates makefile).
- Genprov now patches provider source.
- Logging facility.
- Reference arrays.
- Configure scripts.
- VxWorks Port.
- Darwin Port.
- Solaris Port.
- “Using CIMPLe” Document.



# Demo Outline

- Generate provider skeletons.
- Implement enumerate-instances operation.
- Implement associators operation.
- Implement extrinsic method.
- Install and register provider.
- Verify provider using client tools.

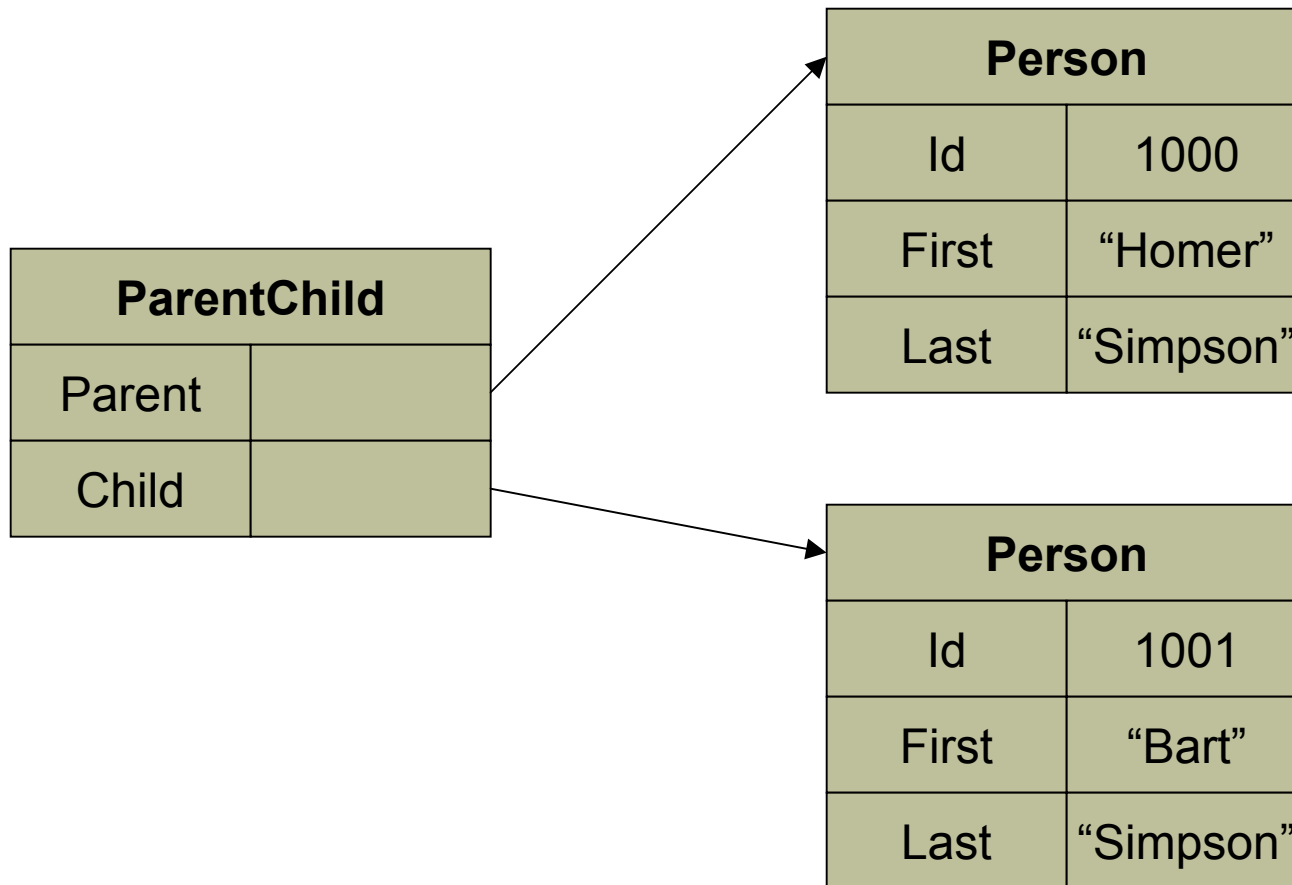
# Demo Classes

```
class Person
{
    [Key] uint32 Id;
    string First;
    string Last;

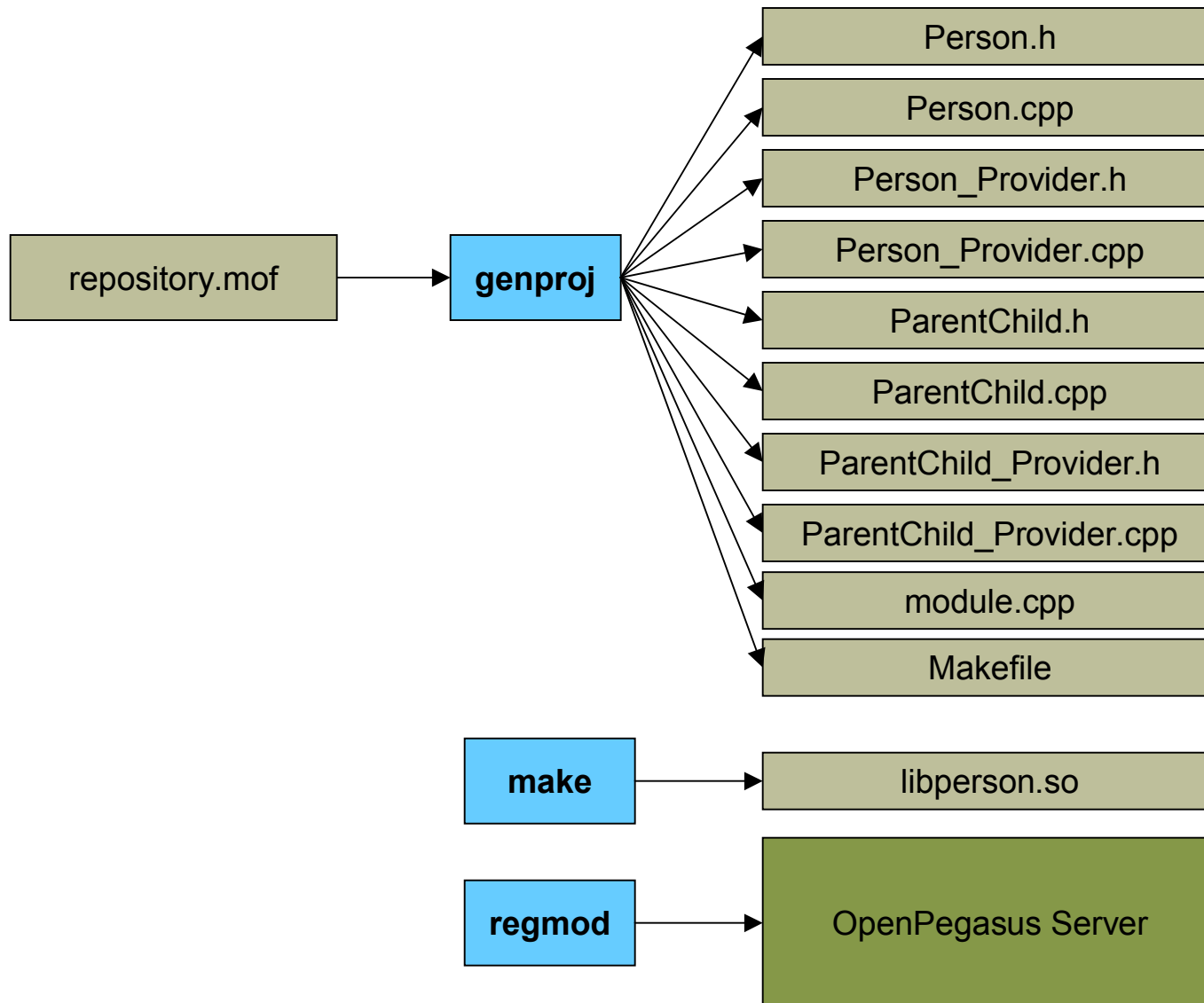
    uint32 GetProperties(
        [In(false), Out] uint32 Id,
        [In(false), Out] string First,
        [In(false), Out] string Last);
};

[Association]
class ParentChild
{
    [Key] Person REF Parent;
    [Key] Person REF Child;
};
```

# Instances for Demo



# Demo Workflow



# Instance Provider

Person\_Provider.cpp

```
Enum_Instances_Status Person_Provider::enum_instances(  
    const Person* model,  
    Enum_Instances_Handler<Person>* handler)  
{  
    Person* p1 = Person::create();  
    p1->Id.value = 1000;  
    p1->First.value = "Homer";  
    p1->Last.value = "Simpson";  
    handler->handle(p1);  
  
    Person* p2 = Person::create();  
    p2->Number.value = 1001;  
    p2->First.value = "Bart";  
    p2->Last.value = "Simpson";  
    handler->handle(p2);  
  
    return ENUM_INSTANCES_OK;  
}
```

Added this

# Association Provider

ParentChild\_Provider.cpp

```
Enum_Instances_Status ParentChild_Provider::enum_instances(  
    const ParentChild* model,  
    Enum_Instances_Handler<ParentChild>* handler)  
{  
    Person* homer = Person::create();  
    homer->Id.set(1000);  
  
    Person* bart = Person::create();  
    bart->Id.set(1001);  
  
    ParentChild* assoc = ParentChild::create();  
    assoc->Parent = homer;  
    assoc->Child = bart;  
  
    handler->handle(assoc);  
  
    return ENUM_INSTANCES_OK;  
}
```

# Extrinsic Method

Person\_Provider.cpp

```
Invoke_Method_Status Person_Provider::GetProperties(  
    const Person* self,  
    Property<String>& First,  
    Property<String>& Last,  
    Property<uint32>& return_value)  
{  
    if (self->Id.value == 1000)  
    {  
        First.set("Homer");  
        Last.set("Simpson");  
        return_value.set(0);  
    }  
    if (self->Id.value == 1001)  
    {  
        First.set("Bart");  
        Last.set("Simpson");  
        return_value.set(0);  
    }  
    return INVOKE_METHOD_OK;  
}
```



# Retargeting the Provider

- To retarget the provider for CMPI:

\$ genmak -C

...

\$ make

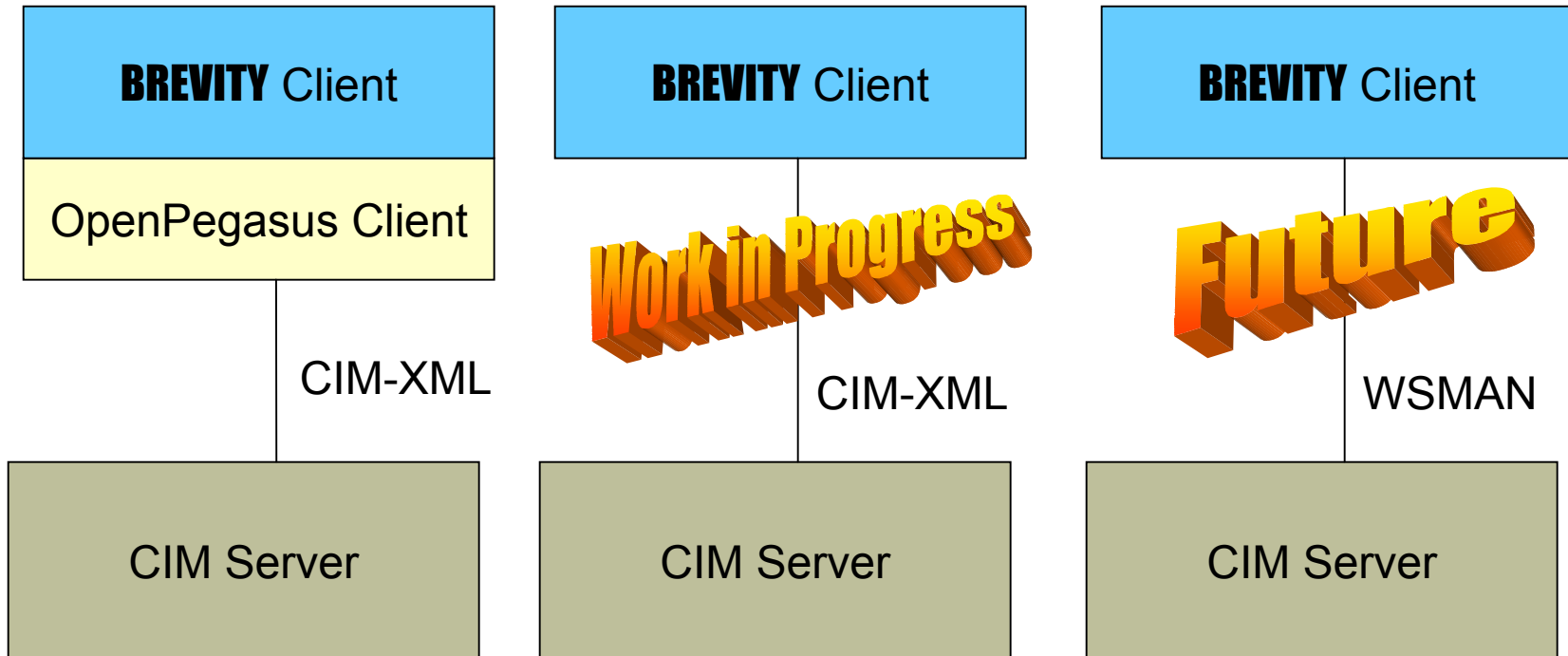


**December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA**

# **BREVITY**

# What is **BREVITY**?

- **BREVITY** is an open-source environment for developing C++ CIM clients.

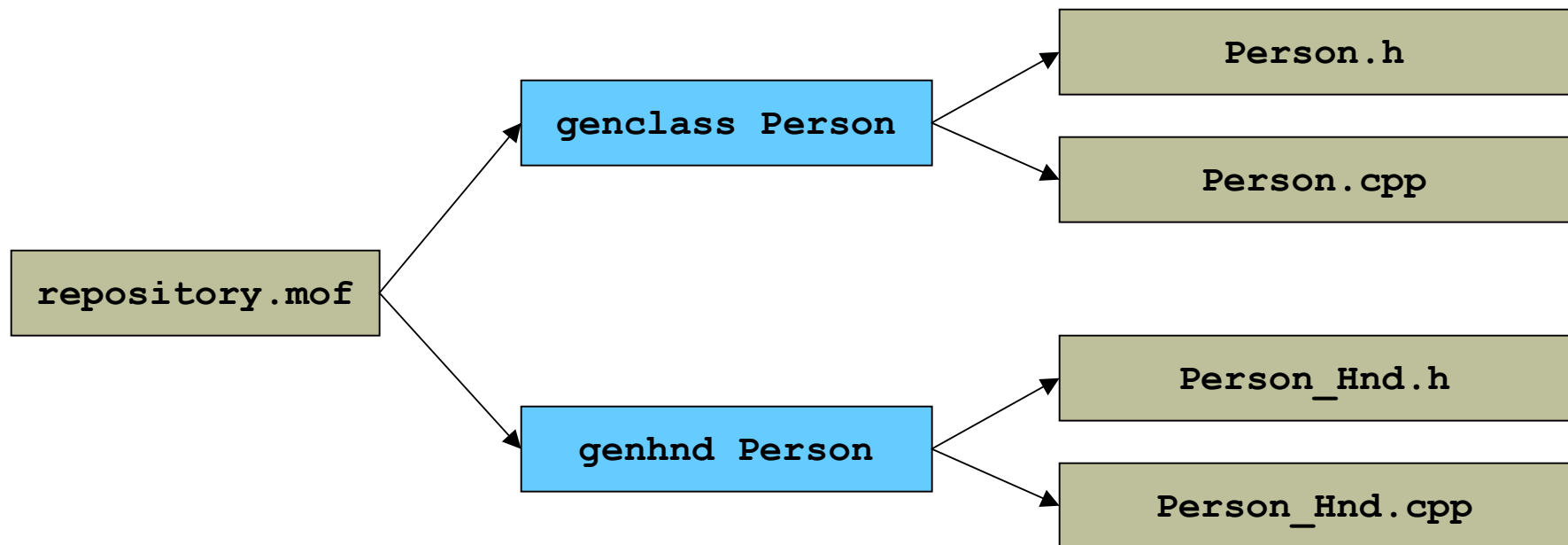




# BREVITY Advantages

- Less development effort and cost.
- Reduced code complexity
- Fewer development errors.
- Support for multiple CIM servers.
- Generates concrete classes.
- Generates extrinsic method stubs.

# Concrete Class Generation





# Working with Generated Class Handles

```
// Create and initialize an instance of Person.  
Person_Hnd h;  
h.Id_value(1000);  
h.First_value("Homer");  
h.Last_value("Simpson");  
h.print();
```



# Working with Generated Class References

```
// Create Person.Id=1000:  
Person_Ref r;  
r.Id_value(1000);
```

# Enumerating Instances

```
Client c;  
c.connect();  
  
Instance_Enum e = c.enum_instances("root/cimv2", Person_Ref());  
  
while (e.more())  
{  
    Person_Hnd person(e.next());  
    person.print();  
}
```



# Invoking an Extrinsic Method with BREVITY

```
Person_Ref ref;  
ref.Id_value(1000);  
  
Arg<String> first;  
Arg<String> last;  
Arg<uint32> result = ref.GetProperties(c, "root/cimv2", first, last);  
  
printf("result=%u:%u\n", result.value(), result.null());  
printf("first=%s:%u\n", first.value().c_str(), first.null());  
printf("last=%s:%u\n", last.value().c_str(), last.null());
```

# Invoking an Extrinsic Method With Pegasus

```
// Build reference for "Person.id=1000"

Array<CIMKeyBinding> keyBindings;
keyBindings.append(CIMKeyBinding("Id", "1000", CIMKeyBinding::STRING));
CIMObjectPath ref(String(), CIMNamespaceName(), "Person", keyBindings);

// Invoke the method:

Array<CIMParamValue> in;
Array<CIMParamValue> out;
CIMValue resultValue = c.invokeMethod(
    "root/cimv2", ref, "GetProperties", in, out);

// Print result:

if (resultValue.getType() != CIMTYPE_UINT32)
{
    // Handle error!
}

UInt32 result;
resultValue.get(result);
printf("result: %u:%u\n", result, resultValue.isNull());

// Check number of output arguments:

if (out.size() != 2)
{
    // Handle error!
}

// Print the output arguments:
```

```
for (UInt32 i = 0; i < out.size(); i++)
{
    const CIMParamValue& param = out[i];

    if (String::equalNoCase(param.getParameterName(), "First"))
    {
        CIMValue firstValue = param.getValue();

        if (firstValue.getType() != CIMTYPE_STRING)
        {
            // Handle error!
        }

        String first;
        firstValue.get(first);
        printf("first: %s:%u\n", (const char*)first.getCString(),
            firstValue.isNull());
    }
    else if (String::equalNoCase(param.getParameterName(), "Last"))
    {
        CIMValue lastValue = param.getValue();

        CIMValue lastValue = param.getValue();

        if (lastValue.getType() != CIMTYPE_STRING)
        {
            // Handle error!
        }

        String last;
        lastValue.get(last);
        printf("last: %s:%u\n", (const char*)last.getCString(),
            lastValue.isNull());
    }
    else
    {
        // Handle error (unknown output parameter)!
    }
}
```



**December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA**

# **Work in Progress**



## Work in Progress

- BREVITY WSMAN client.
- BREVITY over CIMXML, without use of Pegasus client.
- Profile based extensions to CIMPLE and BREVITY.
- WMI Adapter for CIMPLE.
- Self-documented generated source for CIMPLE and BREVITY.
- Improved provider/client tracing tools.

# Questions

