Remedy IT

Your challenge - our solution

AXCIOMA Internals

A LwCCM implementation supporting the IDL to C++11 language mapping

Remedy IT

Your challenge - our solution

Advantages of IDL to C++11

- The IDL to C++11 language mapping has the following advantages compared to the IDL to C++ language mapping
  - Easier to use
  - Language mapping is much safer to use
  - Improved runtime performance
  - Reduced training time
  - Reduced development and test time
  - Faster time to market

Remedy IT

Your challenge - our solution

AXCIOMA Requirements

- LwCCM implementation using the IDL to C++11 Language Mapping
- Existing CIAO deployment plans should be reusable with minimal changes
- Reduced footprint compared to CIAO
- Reduced dependency on CORBA
- Optimized set of features compared to CIAO
  - No support for CCM events
  - No support for the IDL ‘supports’ keyword
  - Component attributes are set to their initial value using a D&C compliant deployment tool but not changeable through CORBA

Remedy IT

Your challenge - our solution

Prerequisites AXCIOMA

- C++ compiler supporting C++11
  - GCC 4.7 or newer
  - Clang version 5 or newer
  - Visual Studio 2015
  - Visual Studio 2017
  - Intel C++ 2016
  - Other C++ compilers have been tested but lack features
- Extensible IDL compiler supporting IDL2, IDL3, and IDL3+
  - RIDL
- Implementation of the IDL to C++11 language mapping for all IDL type constructs
  - TAOX11

Remedy IT

Your challenge - our solution

RIDL

- Ruby based IDL compiler developed by Remedy IT
- Front end with support for IDL2, IDL3, IDL3+, and annotations compatible with DDS-XTypes
- Supports pluggable and extensible backends
- Current available backends
  - IDL to Ruby
  - IDL to C++
  - IDL to C++11
- A LwCCM C++11 backend is developed by extending the IDL to C++11 backend
- Frontend is available as Ruby Gem from http://www.rubygems.org/gems/ridl

Remedy IT

Your challenge - our solution

TAOX11

- Opensource CORBA implementation developed by Remedy IT
- Compliant with the IDL to C++11 language mapping
- Uses TAO core leveraging its portability and features
- Uses RIDL as IDL compiler
- Extended suite of unit tests
- See https://www.taox11.org/

Remedy IT

Your challenge - our solution

AXCIOMA Goals

- Fix the API for component developers by using the IDL to C++11 language mapping
- Independent from the existing CIAO LwCCM implementation
- Existing CIAO deployment plans can be used with minimum changes
- Uses TAOX11 for the C++11 type system and optional CORBA support
- Uses RIDL as extensible IDL compiler

Remedy IT

Your challenge - our solution

DAnCEX11 Requirements

- Be able to deploy AXCIOMA components and connectors
- Provide a flexible and powerful deployment tooling
- Support plugin support for various interception points

Remedy IT

Your challenge - our solution

Requirements DAnCEX11 LM

- A new DAnCEX11 Locality Manager will be developed
- Same set of features as the DAnCE LM
- New internal design and implementation
- Will provide same plugin support as the DAnCE LM

Remedy IT

Your challenge - our solution

AXCIOMA

https://www.axcioma.org

Remedy IT

Your challenge - our solution

AXCIOMA Container Architecture

- AXCIOMA container does not depend on CORBA for its core functionality
- AXCIOMA container is smaller in terms of LOC and footprint compared to CIAO

Remedy IT

Your challenge - our solution

AXCIOMA Container

- AXCIOMA container will provide services to the components hosted
- AXCIOMA installation handlers will create all components
- AXCIOMA architecture is way easier than the CIAO architecture

Remedy IT

Your challenge - our solution

Component CORBA servant

- Signature of the component CORBA servant entry point as used by D&C will be updated with a different return value
- Name of the entry point will be kept the same which means deployment plans don't change
- An AXCIOMA ExecutorLocator is returned which provides access to all needed component related entities:
  - Component and facet executors
  - Component context
  - Component configuration values

Remedy IT

Your challenge - our solution

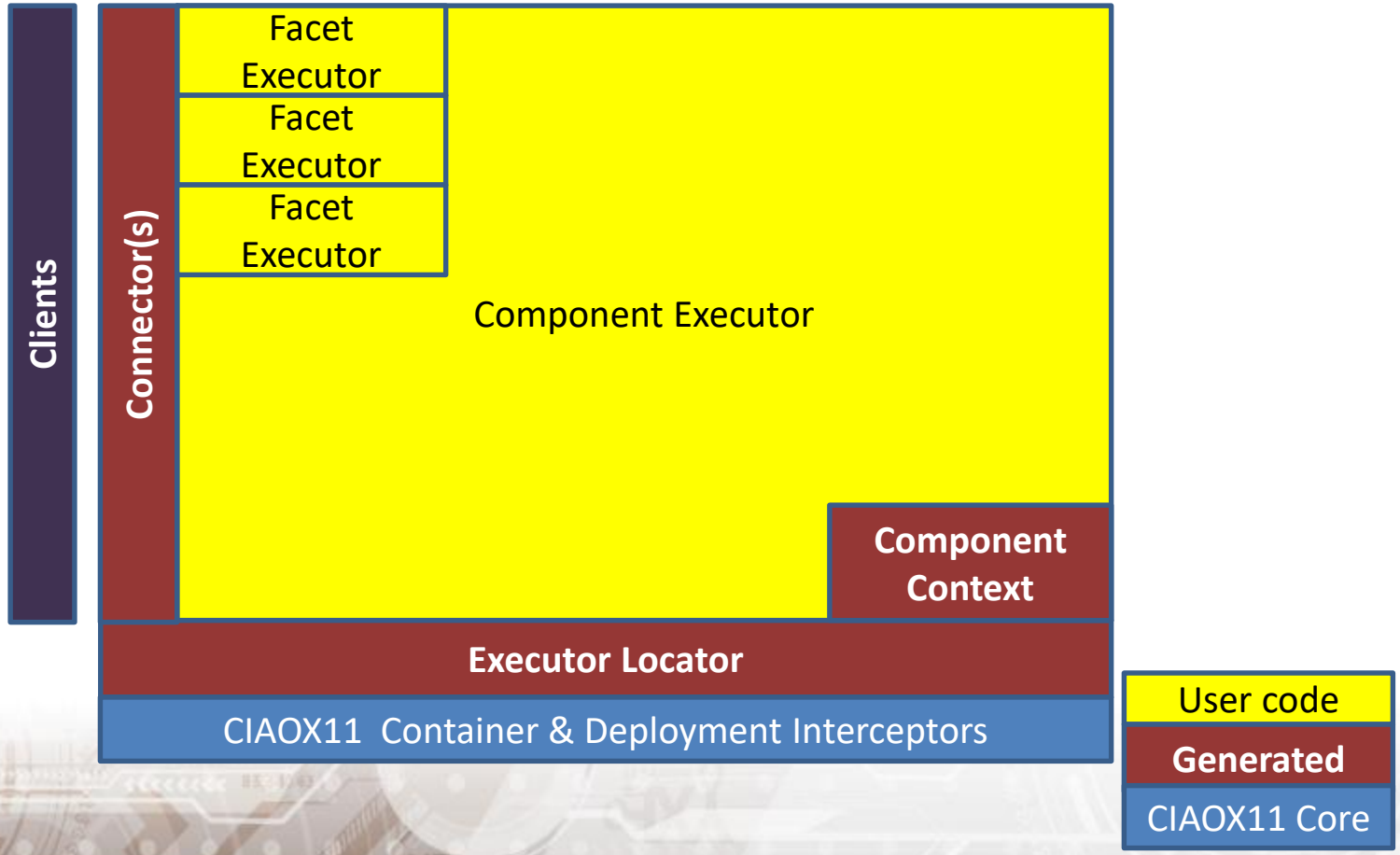
Executor Locator and Configuration Values

- ExecutorLocator is responsible for setting component attributes to their initial value
- Converts attribute names to concrete method invocations onto the user implemented component executor
- Extracts the attribute value from their Any encapsulation

High level architecture

Remedy IT

Your challenge - our solution



Remedy IT

Your challenge - our solution

Component context

- Caches object references for all receptacles of a component
- The CIAO SessionContext `get_CCM_object` is removed
- Simplified API because of the fact that the connection_name is now used as Cookie, no valuetypes anymore!

Remedy IT

Your challenge - our solution

Lifecycle operations

- LwCCM defined lifecycle operations will go directly from the container to the component executor using the ExecutorLocator
- Which operations are called on the component executor is determined by the container, not by the generated code

Remedy IT

Your challenge - our solution

Component CORBA servant

- The component CORBA servant is used by D&C to retrieve CORBA facet object references when remote CORBA interfaces are used
- The component CORBA servant is registered with the POA using its unique D&C id
- Retrieval of facet CORBA object references happens through the POA using the unique component D&C id and the facet name, the references are not cached in the component CORBA servant
- One CORBA component servant implementation will be used for all user components, no user component specific API will be available through CORBA
- The component CORBA servant is not required for the container to operate
- The CORBA servant is handled within CORBA4CCM

Remedy IT

Your challenge - our solution

Connector CORBA servant

- Connectors only provide local facets
- Connectors don’t have CORBA mandatory parts, no CORBA servants are needed
  - A DDS4CCM connector will have no component and facet CORBA servants
  - Leads to a heavy reduction in code generation and footprint for connectors

Remedy IT

Your challenge - our solution

CORBA4CCM connector

- A CORBA4CCM connector is introduced to support request/reply interaction between components
- CORBA4CCM connector delegates all operations and attribute access to the user provided facet executor
- No usage of inheritance for the C++11 facet CORBA servants leads to reduced dependencies and simplified compilation steps

Remedy IT

Your challenge - our solution

DDS4CCM

- DDS4CCM connectors are implemented using C++11
- Use a similar C++ templated connector framework as CIAO
- Optimal integration requires a DDS vendor that natively supports IDL to C++11 for
  - Built in DDS entities and types
  - User defined types as generated by TAOX11
  - IDL defined type specific DataReader and DataWriter

Remedy IT

Your challenge - our solution

RTI DDS integration

- RTI DDS lacks support for IDL to C++11
- AXCIOMA will provide C++11 API to the user and convert internally to the RTI C++ API
  - Makes the integration possible
  - Can be adapted for other DDS vendors that lack C++11 support
  - Templated design allows for full optimization when a DDS vendor with native IDL to C++11 support is integrated
- No usage of the RTI CORBA Compatibility Kit (CCK)
- Runtime conversion of user data to the RTI C++ types
- New set of RIDL backends for generating the type conversion

Remedy IT

Your challenge - our solution

AMI4CCM

- AMI4CCM connectors are implemented using C++11 and generated by RIDL
- TAOX11 CORBA AMI support is used
- CORBA AMI support is completely hidden inside the AMI4CCM connector, no CORBA AMI is exposed to the component developer

Remedy IT

Your challenge - our solution

Timer Support

- Timer support is provided by the TT4CCM connector
- TT4CCM shields application code from low level middleware details
- AXCIOMA does provide access to the TAOX11 ORB through the LwCCM service registry and the TAOX11 ORB will provide access to its ACE Reactor
- Future revisions of the LwCCM/UCM standard need to address the lack of a timer concept in LwCCM

Remedy IT

Your challenge - our solution

Valuetypes

- AXCIOMA modifies the LwCCM IDL to not use valuetypes
  - ConfigValue are changed to an IDL struct
  - Cookie will be an IDL string simplifying the usage of it
- Not using valuetypes reduces footprint and complexity

Remedy IT

Your challenge - our solution

Starter Executor Generation (1)

- AXCIOMA uses RIDL for generating the starter code for the component and facet executors
- Unique RIDL regeneration blocks will be added to the generated starter code
  - A regeneration block is enclosed by a begin and end marker
  - Marker is unique for the file where it is used in
- RIDL will read in the existing file, store the blocks and place them back upon regeneration

Remedy IT

Your challenge - our solution

Starter Executor Generation (2)

```
// Some example markers, the @@{__RIDL_REGEN_MARKER__}  part can be changed through the commandline
// flags of RIDL

//@@{__RIDL_REGEN_MARKER__} - BEGIN : Shapes_Receiver_Impl[src_includes]
#include "ace/OS_NS_time.h"
//@@{__RIDL_REGEN_MARKER__} - END : Shapes_Receiver_Impl[src_includes]

void
info_out_data_listener_exec_i::on_one_data (const ::ShapeType& datum, const ::CCM_DDS::ReadInfo&
     info) override
{
  //@@{__RIDL_REGEN_MARKER__} - BEGIN : info_out_data_listener_exec_i::on_one_data[_datum_info]
  std::cout << "Received shape <" << datum << std::endl;
  //@@{__RIDL_REGEN_MARKER__} - END : info_out_data_listener_exec_i::on_one_data[_datum_info]
}

Receiver_exec_i::~Receiver_exec_i ()
{
  //@@{__RIDL_REGEN_MARKER__} - BEGIN : Shapes_Receiver_Impl::Receiver_exec_i[destructor]
  //@@{__RIDL_REGEN_MARKER__} - END : Shapes_Receiver_Impl::Receiver_exec_i[destructor]
}
```

AXCIOMA Release

Remedy IT

Your challenge - our solution

AXCIOMA V2.0

- Android, RHEL 6.x/7.x, Fedora >= 17, OpenSuSE >= 12.2, CentOS 6.x/7.x, and Windows 32/64 using MinGW, Visual Studio 2015, and Intel C++ 2016 as supported platforms
- Support for components and connectors
- CORBA support through CORBA4CCM
- DDS4CCM Event and State connectors using RTI Connext DDS 6.0.0
- AMI4CCM support
- TT4CCM support
- Full suite of unit tests and examples
- Full D&C compliant toolchain
- Full documentation

Resources

Remedy IT

Your challenge - our solution

External links

- TAOX11
  - https://taox11.remedy.nl/
- IDL to C++11 specification
  - http://www.omg.org/spec/CPP11

Remedy IT

Your challenge - our solution

Want to know?

- Go to our AXCIOMA website at https://www.axcioma.com
- See our other presentations at http://www.slideshare.net/RemedyIT

Remedy IT

Your challenge - our solution

Contact

Remedy IT
The Netherlands

tel.: +31(0)88 – 053 0000
e-mail: sales@remedy.nl
website: www.remedy.nl

Twitter: @RemedyIT
Slideshare: RemedyIT