CoSMIC: An MDA Tool Suite for Application Deployment and Configuration

Tao Lu, Emre Turkay, Aniruddha Gokhale*, Douglas Schmidt
Institute for Software Integrated Systems
Vanderbilt University, Nashville TN
*Contact Author: a.gokhale@vanderbilt.edu

1. Overview

The Component Synthesis using Model Integrated Computing (CoSMIC) [1] is the Object Management Group (OMG)'s Model Driven Architecture (MDA)[2] standards-based tool suite targeted primarily for distributed real-time and embedded (DRE) component-based applications.

The OMG's Deployment and Configuration (D&C) Specification [3], which was recently adopted, describes the mechanisms by which distributed component-based applications are configured and deployed. This paper describes how our CoSMIC MDA tool suite has been tailored to address the requirements of the D&C specification.

The D&C specification outlines the following steps in the deployment process:

1. *Packaging* – which involves bundling a suite of software binary modules and metadata representing application components, where a component can be monolithic (standalone) or an assembly of subcomponents
2. *Installation* – which provides populating a repository with the packages required by the application
3. *Configuration* – which includes configuring the packages with the appropriate parameters to satisfy the functional and systemic requirements of application without constraining to any physical resources
4. *Planning* – which comprises making appropriate deployment decisions, such as identifying the entities, such as CPUs, of the target environment where the packages will be deployed
5. *Preparation* – which involves moving the binaries to the identified entities of the target environment
6. *Launching* – which involves triggering the installed binaries and bringing the application to a ready state

The CoSMIC MDA tool suite has to date addressed the packaging and configuration aspects of the deployment process outlined above. These aspects are described in detail below.

2. CoSMIC's Component Assembly and Deployment Modeling Language

Applications based on component middleware are composed from instantiation of different component types, which are partitioned, assembled and then deployed. For applications, particularly in the DRE domain, the MDA tool must ensure that the composition strategies provide optimum resource utilization and deliver the required quality of service (QoS) guarantees to the application.

The Component Assembly & Deployment Modeling Language (CADML) is developed as part of the CoSMIC MDA tool suite targeting the assembly and deployment aspect of the DRE application deployment process.

The CADML, based on the meta-model illustrated in Figure 1, is a visual language tool developed using the Generic Modeling Environment (GME) framework [4]. Visual languages developed using GME benefit from the following features:

- GUI interface supporting all general GUI application features with very generic semantic mapping.
- Library importing and exporting capability.
- Type system defined in the meta-model, which supports inheritance and instantiation. This introduces object-oriented design (OOD) in the modeling paradigm.
- Formalized constraints specified in the meta-model to validate the model.
- Plug-in of analysis and synthesis tools that interpret the models

The current release of CoSMIC's CADML tool supports the OMG CORBA[5] Component Model (CCM)[6] standard and works out of the box with the Component Integrated ACE ORB (CIAO)[7], which is a real-time CCM implementation we are developing.

The CADML modeling paradigm allows CIAO-based DRE application developers to model the component assembly comprising the connections between different components of the applications.

The CADML interpreter synthesizes component assembly metadata as XML descriptors that are used by the middleware deployment tools. Different descriptor files represent different application scenarios. With the support of general component repository, an application developer could build up different application scenarios by only providing the specific descriptors.

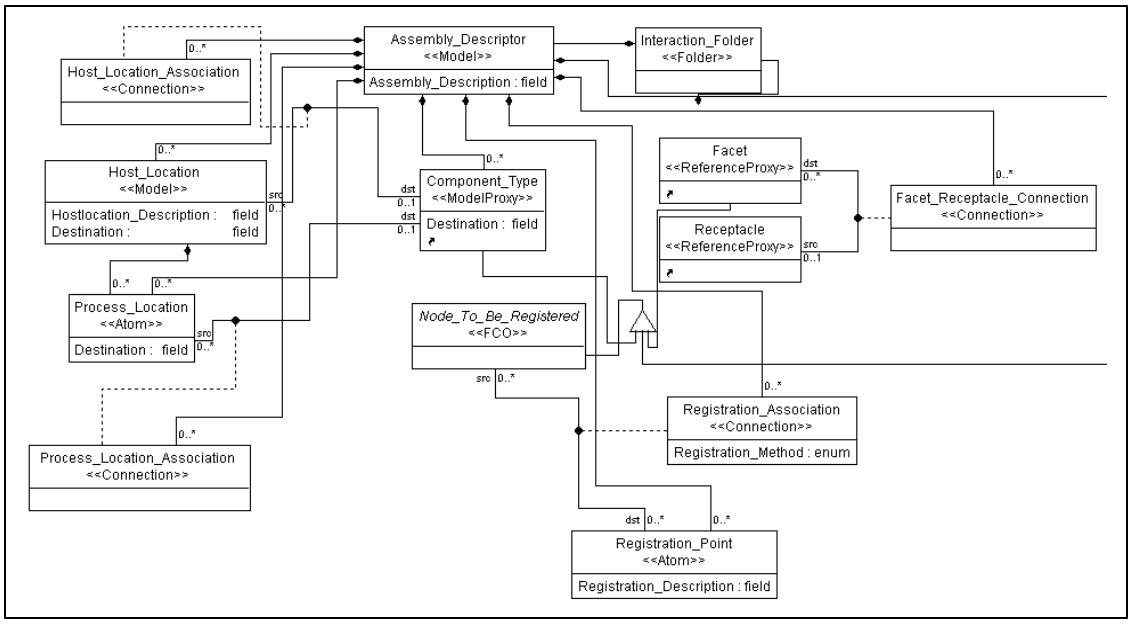

Figure 1: CADML Visual Paradigm

Figure 2 illustrates an example avionics assembly modeled using the CADML visual modeling paradigm. This example illustrates a trivial application comprising a timer driven global positioning (GPS) component that drives the aircraft's airframe and pilot's navigation display.

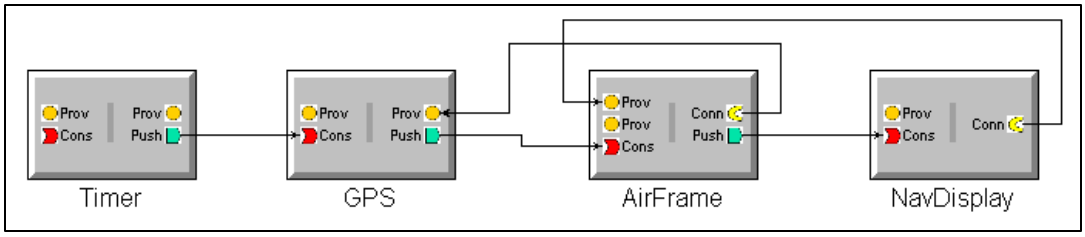

Figure 2: Modeling an Avionics Application Assembly

3. CoSMIC's Options Configuration Modeling Language

Most applications today increasingly are built using commercial off-the-shelf (COTS) middleware. These COTS middleware must be customized to suit the application's functional and non-functional requirements. This customization can be achieved by enabling a combination of middleware configuration options chosen carefully from among a large set of such tunable options provided by the middleware. Each option addresses different performance aspects of applications. As a result some options are mutually dependent while certain combinations of options often conflict with each other. The task of choosing the right set of compatible options thus cannot be done in an *ad hoc* way but instead must be done by a tool.

CoSMIC's Options Configuration Modeling Language (OCML) is another visual modeling paradigm developed using the GME framework to address the configuration aspect of the deployment process. It provides a language to define the constraints and dependencies of the options that can then be used to customize the middleware for DRE applications.

Error! Reference source not found. illustrates the OCML MDA development process. In its current form, OCML supports option configuration and validation for the TAO real-time CORBA ORB [8], which is used as the ORB layer for the CIAO CCM implementation.

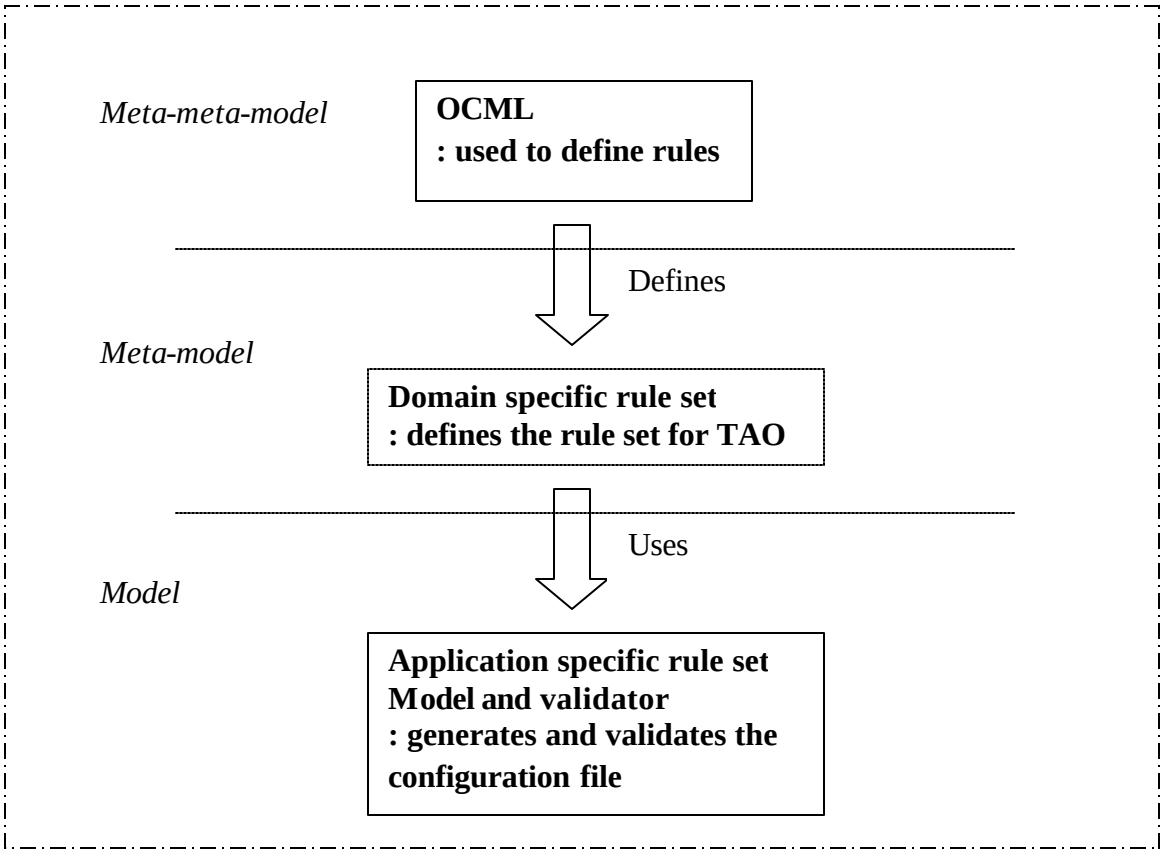

Figure 3: MDA Developing Process in OCML

The OCML paradigm has been developed for middleware developers who can use it to model the dependency and compatibility rules between various options.

Figure 4 illustrates an example of an options compatibility and dependency rule modeled by a TAO ORB developer. This rule showcases an *AND expression* used to model a dependency of an option on more than one option simultaneously

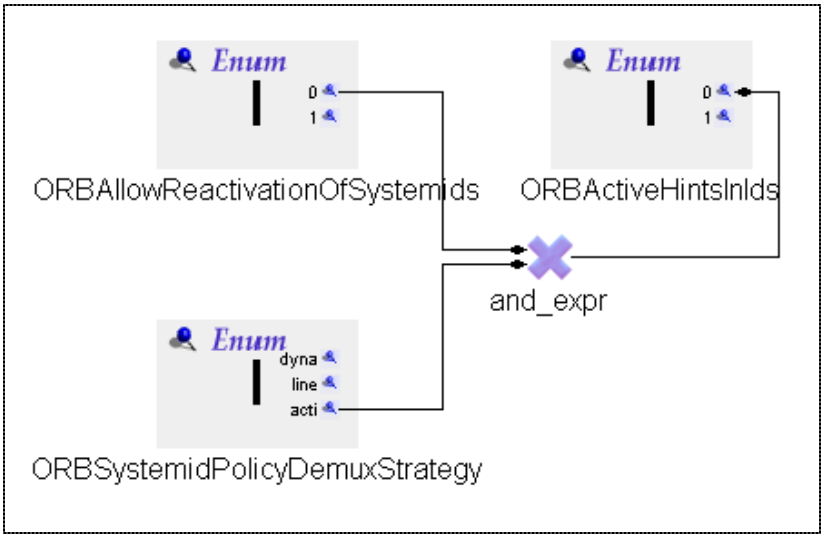

Figure 4: OCML Configuration Rule for TAO ORB

The OCML meta-model is used with two different purposes. One purpose serves the validation of pre configured options, while other allows interactive option file construction.

A validator checks an application-supplied TAO ORB configuration file against the OCML modeled rules for TAO ORB. If a rule check fails, the application developer is informed. Alternately, an application developer can choose to use a graphical user interface tool that accompanies the OCML paradigm. This framework provides an interactive environment to application developers to build a set of compatible options. This interactive tool consults the rules modeled by the middleware developer thereby constraining the choice of options to a valid, compatible set.

A sample TAO ORB options configuration file in XML is shown below:

```
<?xml version='1.0'?>
<!-- Converted from ./performance-tests/Latency/DII/svc.conf by
svcconf-convert.pl -->
<ACE_Svc_Conf>
 <!--  -->
 <!--  -->
 <static id="Advanced_Resource_Factory" params="-ORBresources global -
ORBReactorMaskSignals 0 -ORBInputCDRAllocator null -ORBReactorType
select_st -ORBConnectionCacheLock null"/>
 <static id="Server_Strategy_Factory" params="-ORBPOALock null -
ORBAllowReactivationOfSystemids 0 -ORBActiveHintsInIds 1"/>
 <static id="Client_Strategy_Factory" params="-ORBProfileLock null -
ORBClientConnectionHandler RW"/>
<static id="Resource_Factory" params="-ORBConnectionPurgingStrategy
null -ORBConnectionCacheMax 1024"/>
</ACE_Svc_Conf>
```

An XML file of this kind can be built using the interactive tool or a pre configured file can validated using the rules engine provided by OCML.

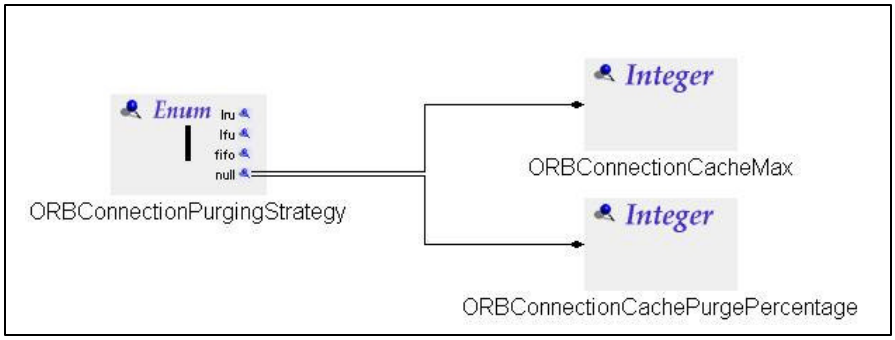

Figure 5: TAO ORB Options

The OCML models for middleware options are translated into XML by OCML interpreters. The synthesized XML for Figure 4 and Figure 5 are shown below:

```
 <RULE name="ActiveHints_Rule">
    <IMPLIES>
      <VALUE category="Server_Strategy_Factory"
option="ORBAllowReactivationOfSystemids">
        0
      </VALUE>
      <NOT>
        <SELECT category="Server_Strategy_Factory"
option="ORBActiveHintsInIds"/>
      </NOT>
    </IMPLIES>
  </RULE>

  <RULE name="NoConnectionPurging_Rule" category="Resource_Factory">
    <AND>
      <IMPLIES>
        <VALUE option="ORBConnectionPurgingStrategy">
          null
        </VALUE>
        <AND>
          <NOT>
            <SELECT option="ORBConnectionCacheMax"/>
          </NOT>
          <NOT>
            <SELECT option="ORBConnectionCachePurgePercentage"/>
          </NOT>
        </AND>
      </IMPLIES>
    </AND>
  </RULE>
```

4. Work in progress

The CoSMIC tool suite is currently being augmented to address the planning aspect of the deployment process. In particular, we are developing suitable paradigms for expressing the QoS requirements of DRE applications. Analyzing the QoS requirements will provide the basis for making the planning decisions, such as identifying the entities in the target environment where different parts of the application will be deployed. These QoS requirements will also drive some of the assembly and configuration aspects thereby interworking with the CADML and OCML tools. Moreover, OCML has to be refined to capture option configurations at four distinct layers within the CCM. For example, configuration of a middleware, such as CIAO CCM, involves configuring parameters at several layers including the ORB, the component server, the containers, and the components. Some of the configuration decisions at all these layers will need lazy

evaluation at planning time when the QoS requirements are analyzed in the context of the target environment.

5. References

[1] Aniruddha S. Gokhale, Douglas C. Schmidt, Tao Lu, Balachandran Natarajan, Nanbor Wang: CoSMIC: An MDA Generative Tool for Distributed Real-time and Embedded Applications. *Middleware Workshops* 2003: 300-306

[2] Object Management Group: Model Driven Architecture(MDA)Document number ormsc/2001-07-01 Architecture Board ORMSC1July 9, 2001

[3] Object Management Group: Deployment and Configuration of Component-based Distributed Applications, *An Adopted Specification of the Object Management Group, Inc.*June 2003 Draft Adopted Specification ptc/July 2002

[4] Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason IV C., Nordstrom G., Sprinkle J., Volgyesi P.: The Generic Modeling Environment, Workshop on Intelligent Signal Processing, accepted, Budapest, Hungary, May 17, 2001.

[5] Object Management Group, The Common Object Request Broker: Architecture and Specification, 2.6 edition, Dec. 2001.

[6] BEA Systems, et al., CORBA Component Model Joint Revised Submission, *Object Management Group*, OMG Document orbos/99-07-01 edition, July 1999.

[7] Nanbor Wang, Douglas C. Schmidt, Aniruddha Gokhale,Christopher D. Gill, Balachandran Natarajan, Craig Rodrigues, Joseph P. Loyall, and Richard E. Schantz, "Total Quality of Service Provisioning in Middleware and Applications," *Elsevier Journal of Microprocessors and Microsystems*, vol. 26, no. 9-10, Jan 2003.

[8] Douglas C. Schmidt, Aniruddha Gokhale, Balachandran Natarajan, et al, " TAO: A Pattern-Oriented Object Request Broker for Distributed Real-time and Embedded Systems," IEEE Distributed Systems Online, Feb 2002.