An Integrated Aspect-oriented Model-driven Development Toolsuite for Distributed Real-time and Embedded Systems

Aniruddha S. Gokhale
ISIS, Dept. of EECS
Vanderbilt University
Nashville, TN 37235
a.gokhale (at) vanderbilt.edu

Jeffrey G. Gray
Dept. of CIS
University of Alabama at Birmingham
Birmingham, AL 35294
gray (at) cis.uab.edu

Abstract:

Model-driven development (MDD) has been gaining importance as an approach to resolving lifecycle challenges of large-scale distributed real-time and embedded (DRE) systems (e.g., joint emergency response systems, avionics, and automotive systems), including design, development, testing, maintenance and evolution. DRE systems are characterized by their stringent requirements for quality of service (QoS), such as predictable end-to-end latencies, timeliness and scalability. Delivering the QoS needs of DRE systems entails the need to configure correctly, fine tune and provision the infrastructure used to host the DRE systems, which crosscuts different layers of middleware, operating systems and networks. Addressing these tangled deployment and configuration concerns of DRE systems requires integrating the principles of Aspect-Oriented Software Design (AOSD) with MDD.

This paper describes the integration of the C-SAW aspect weaving tool with the CoSMIC MDD tool suite to resolve the accidental complexities involved in the configuration and deployment of component middleware-based DRE systems.

1. Challenge Problems

Quality of service (QoS)-enabled component middleware, such as lightweight CORBA Component Model (CCM) (and to some extent J2EE and .Net for the domains they are used in) are increasingly being used to develop and control mission-critical, large-scale DRE systems. Figure 1 illustrates a representative DRE system highlighting a joint emergency response system. DRE systems share the following characteristics giving rise to tangled concerns in its development and maintenance lifecycle:

1. **Heterogeneity.** Large-scale DRE systems often run on a variety of computing platforms that are interconnected by different types of networking technologies with varying QoS properties. The efficiency and predictability of DRE systems built using different infrastructure components varies according to the type of computing platform and interconnection technology.
2. **Deeply embedded properties**. DRE systems are frequently composed of multiple embedded subsystems. For example, an anti-lock braking software control system forms a resource-constrained subsystem that is part of a larger DRE application controlling the overall operation of an automobile.
3. **Simultaneous support for multiple QoS properties**. DRE software controllers are increasingly replacing mechanical and human control of critical systems. These controllers

must simultaneously support many challenging QoS constraints, including (1) real-time requirements, such as low latency and bounded jitter, (2) availability requirements, such as fault propagation/recovery across distribution boundaries, (3) security requirements, such as appropriate authentication and authorization, and (4) physical requirements, such as limited weight, power consumption, and memory footprint. For example, a distributed patient monitoring system requires predictable, reliable, and secure monitoring of patient health data that can be distributed in a timely manner to healthcare providers.

Figure 1. Sample DRE System

4. **Large-scale, network-centric operation**. The scale and complexity of DRE systems makes it infeasible to deploy them in disconnected, standalone configurations. The functionality of DRE systems is therefore partitioned and distributed over a range of networks. For example, an urban bio-terrorist evacuation capability requires highly distributed functionality involving networks connecting command and control centers with bio-sensors that collect data from police, hospitals, and urban traffic management systems.
5. **Dynamic operating conditions.** Operating conditions for large-scale DRE systems can change dynamically, resulting in the need for appropriate adaptation and resource management strategies for continued successful system operation. In civilian contexts, for instance, power outages underscore the need to detect failures in a timely manner and adapt in real-time to maintain mission-critical power grid operations. In military contexts, likewise, a mission mode change or loss of functionality due to an attack in combat operations requires adaptation and resource reallocation to continue with mission-critical capabilities.

2. Concern Separation using Model-Driven Development

MDD technologies, such as the Object Management Group (OMG)'s Model-driven Architecture (MDA), have emerged to address the different lifecycle challenges of DRE systems outlined above, which includes satisfying both the functional and QoS needs of DRE systems during design, development, testing, maintenance, and evolution stages. Figure 2 illustrates how a MDD tool chain coordinates with design-time analysis tools and run-time infrastructure (such as middleware, OS and networks) to address DRE system lifecycle challenges described below:

- *Specification and implementation*, which enables application functionality specification, partitioning, and implementation as components.
- *Packaging*, which allows bundling a suite of software binary modules and metadata representing application components.
- *Installation*, which involves populating a repository with the packages required by the application.
- *Configuration*, which allows configuration of the packages with the appropriate parameters to satisfy the functional and systemic requirements of application without constraining to any physical resources.

- *Planning*, which makes appropriate deployment decisions including identifying the entities, such as CPUs, of the target environment where the packages will be deployed.
- *Preparation*, which moves the binaries to the identified entities of the target environment.
- *Launching*, which triggers the installed binaries and brings the application to a ready state.
- *Adaptation*, which enables run-time reconfiguration and resource management to maintain end to-end QoS.

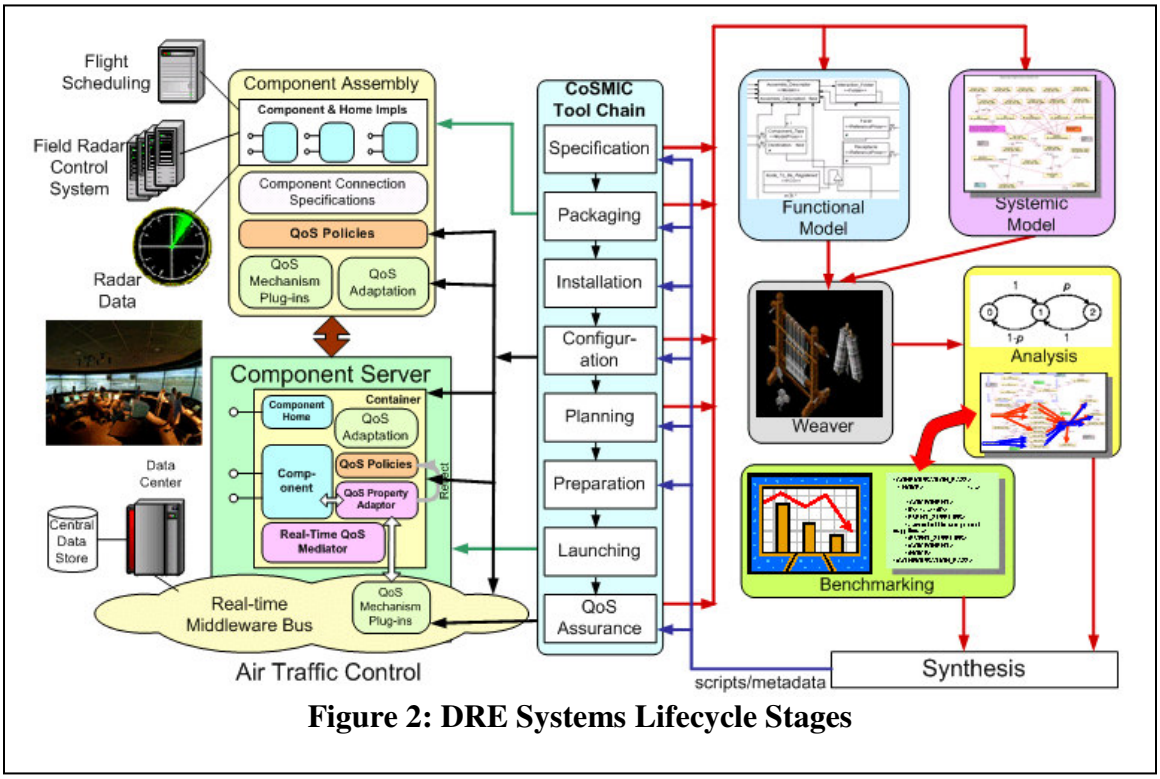

Figure 2: DRE Systems Lifecycle Stages

Figure 2 also demonstrates how the various stages of the DRE systems lifecycle are tangled with different layers of the infrastructure that host the DRE systems. MDD tools in association with AOSD techniques can assist in untangling the crosscutting concerns at each stage of the DRE lifecycle, which helps improve productivity and time-to-market.

3. Applying AOSD Principles to MDD Toolsuites

This section showcases how the CoSMIC MDD tool suite is used in conjunction with the C-SAW model weaver to address the tangled concerns encountered in the DRE system lifecycle. We briefly describe the two tools and show how we are integrating them to resolve DRE system lifecycle challenges.

3.1 The CoSMIC MDD Tool suite

The deployment and configuration of middleware typically involves manual modification to meta-data that is split across multiple XML descriptors. A key evolution and change management

problem exists because of the inter-dependencies and crosscutting between descriptors, and the fact that manual modification of large XML files is error-prone.

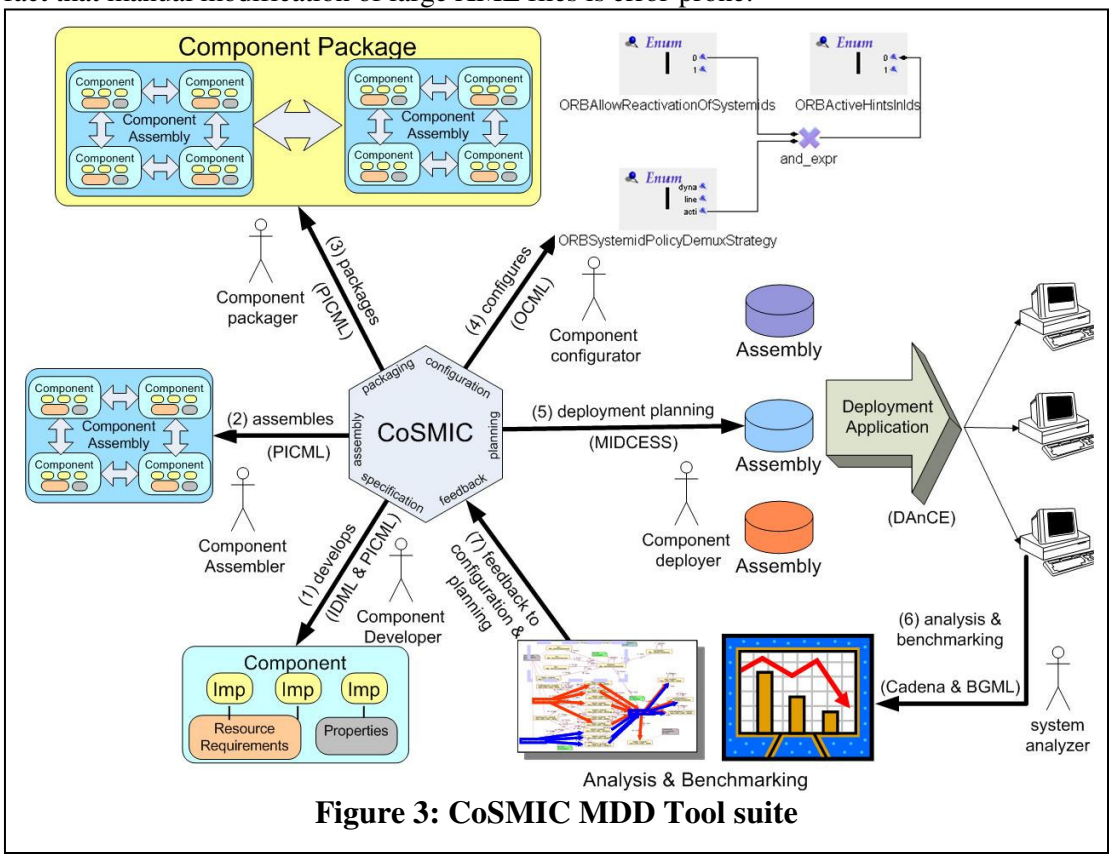

Figure 3: CoSMIC MDD Tool suite

The *Component Synthesis with Model Integrated Computing* (CoSMIC) [3] tools (available at http://www.dre.vanderbilt.edu/cosmic /) are developed using the Generic Modeling Environment (GME) [6], which is a metamodeling environment that defines the modeling paradigms for each stage of the CoSMIC tool chain. The CoSMIC MDD toolsuite illustrated in Figure 3 is a collection of domain-specific modeling languages (DSMLs) and generative tools to address the concerns at different stages of DRE systems lifecycle shown in Figure 2.

The CoSMIC tools use GME to enforce their "correct by construction" techniques, as opposed to the "construct by correction" techniques commonly used by post-construction tools, such as compilers, source-level debuggers, and script validators. CoSMIC ensures that the rules of construction – and the models constructed according to these rules – can evolve together over time. Each CoSMIC tool synthesizes metadata in XML for use in the underlying middleware. The CoSMIC tool suite currently uses a platform-specific model approach that integrates the modeling technology with QoS-enabled component middleware, such as CIAO [1] (available at http://www.cs.wustl.edu/~schmidt/CIAO.html). CoSMIC provides the capability to inter work with third party model checking tools, such as Cadena from Kansas State University [5], and aspect model weavers, such as C-SAW [4].

3.2 Aspect Weaving at the Modeling Level

We have discovered that many properties and policies within a DRE system (e.g., eager/lazy evaluation, processor assignment policies, component placement, and QoS policies) crosscut the

model hierarchy and are spread across many locations within a model. These crosscutting properties represent a common global concern, but their scattering across the model requires much manual adaptation in order to explore design variations. Extensive manual adaptation within a modeling tool can be time consuming and an error prone activity. In combination with CoSMIC, aspect modeling approaches offer relevant benefits of applying aspect-oriented principles to models in order to better isolate crosscutting concerns. The Constraint-Specification Aspect Weaver (C-SAW) model weaver (available at http://www.cis.uab.edu/gray/Research/C-SAW/ and shown in Figure 4) is a general model transformation engine that provides a capability to describe the essence of a concern and transform a model accordingly. In C-SAW, aspects are defined at the modeling abstraction level using the Embedded Constraint Language (ECL). C-SAW provides the ability to assist a modeler in quickly inserting and removing new properties and policies into a model without the need for extensive manual adaptation.

3.3 Brief Example Application of AOSD in CoSMIC

EQAL[1] (Event Quality Aspect Language) is a domain-specific modeling language within CoSMIC that assists in the specification and configuration of federated event channels [2]. Figure 4 shows an EQAL model with three sites. To scale-up an event channel requires changes to many different model locations. C-SAW can automate the scaling task - an ECL strategy specification is used to scale up any site as well as the corresponding connections in the EQAL model. C-SAW takes the original EQAL model and the ECL specifications, and then generates the new scaled-up EQAL model with additional sites. Three steps are included, as listed below and show in Listing 1:

- Add extra CORBA_Gateways to the existing sites
- Repeatedly replicate the site as an instance
- Create connections between all of the sites

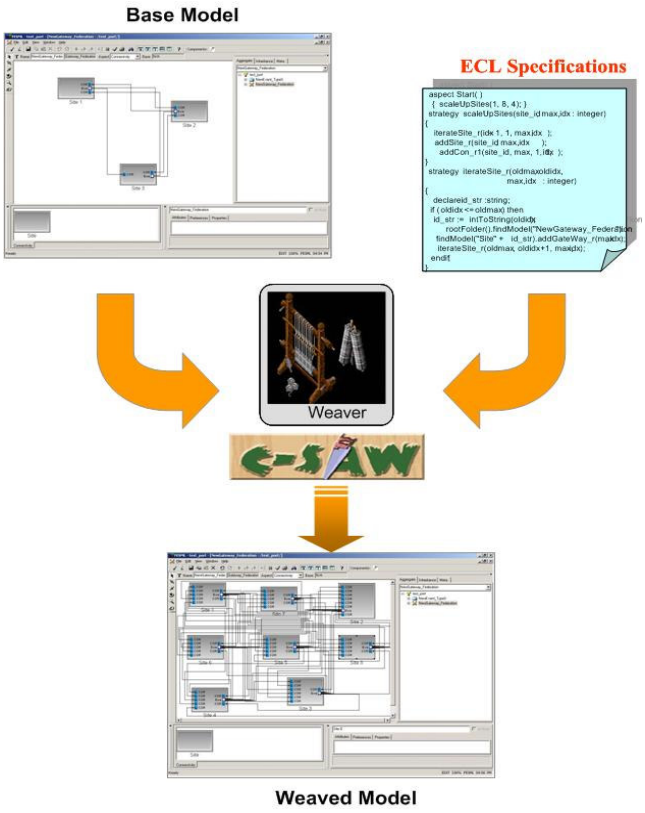

Figure 4: Application of C-SAW

[1] Original name for this tool was FESML but was later subsumed within the EQAL tool in CoSMIC

```
//recursively add sites
strategy addSite_r(max,idx: integer)
{
  if (idx <= max) then
    addSite(idx);
    addSite_r(max,idx+1);
  endif;

  rootFolder().findModel("NewGateway_Federation").
               findModel("Site 1").addGateWay_r(max, idx);
}

strategy addSite(idx: integer)
{
  // update NewGateWay_Federation model via adding one Site instance
  rootFolder().findModel("NewGateway_Federation").updateNGF(idx);
}
                     strategy updateNGF(idx:integer)
{
  declare site, site_ins : object;
  declare id_str : string;

  id_str := intToString(idx);

  // create Site instance
  site := findModel("Site 1");
  site_ins := addInstance("Site", "Site " + id_str, site);
}

strategy addGateWay_r(max, idx: integer)
{
  if (idx<=max) then
      addGateWay();
      addGateWay(max, idx+1);
  endif;
}

strategy addGateWay( )
{
  addAtom("CORBA_Gateway", "CORBA_Gateway");
}
```

Listing 1: ECL Specification to Scale EQAL Model

We have applied the strengths of the C-SAW in a similar fashion to different CoSMIC DSMLs. For example, we have used C-SAW to scale the models of component assemblies and packages that are modeled using CoSMIC's PICML tool, and are currently exploring its use to define other crosscutting concerns, such as configuring the component middleware's container QoS policies, or the component collocation and placement strategies.

4. Conclusions

This paper illustrates the tangling of concerns in the deployment and configuration of distributed real-time and embedded systems. Model driven generative technologies help address these concerns by alleviating several accidental complexities arising in the modeling process. Yet, MDD tools alone are not sufficient since they cannot scale in some cases. Additionally, some of

the modeling activities can become tedious and repetitive while addressing crosscutting concerns. Aspect weaving at the modeling level resolves these problems. This paper describes ongoing work along with a short case study on integrating the C-SAW aspect weaving tool with the CoSMIC model driven development tool suite. All the tools described in this paper are available for download at the web sites referenced in the paper.

References:

1. *Component Integrated ACE ORB*, http://www.cs.wustl.edu/~schmidt/CIAO.html
2. Gan Deng, Aniruddha Gokhale and Balachandran Natarajan, "Model-Driven Integration of Federated Event Services in Real-Time Component Middleware," *Proceedings of the 42nd ACM Southeast Conference*, Huntsville, AL, April 2-3, 2004, pp. 353-356.
3. Aniruddha Gokhale, Douglas Schmidt, Balachandran Natarajan, Jeff Gray, and Nanbor Wang, "Model-Driven Middleware," in *Middleware for Communications*, (Qusay Mahmoud, ed.), John Wiley and Sons, 2004, Chapter 7, pp. 163-187.
4. Gray, J., Bapty, T., Neema, S., and Tuck, J., "Handling Crosscutting Constraints in Domain-Specific Modeling," *Communications of the ACM*, October 2001, pp. 87-93.
5. John Hatcliff, William Deng, Matthew Dwyer, Georg Jung, Venkatesh Prasad Ranganath, "Cadena: An Integrated Development, Analysis, and Verification Environment for Component-based Systems," *Proceedings of the 2003 International Conference on Software Engineering (ICSE 2003)*, Portland, Oregon, May 2003, pp. 160-173.
6. Lédeczi Á., Bakay A., Maroti M., Volgyesi P., Nordstrom G., Sprinkle J., Karsai G , "Composing Domain-Specific Design Environments," *IEEE Computer*, November 2001, pp. 44-51.