Cadena: An Integrated Development Environment for Analysis, Synthesis, and Verification of Component-based Systems*

Adam Childs, Jesse Greenwald, Venkatesh Ranganath, Xianghua Deng, Matthew Dwyer, John Hatcliff, Georg Jung, Prashant Shanti, Gurdip Singh

Department of Computing and Information Sciences
Manhattan, KS 66506, USA
`http://cadena.projects.cis.ksu.edu`

Abstract. This tool paper gives an overview of Cadena – an integrated environment for building and modeling systems built using the CORBA Component Model (CCM). Cadena provides facilities for defining component types using CCM IDL, specifying dependency information and transition system semantics for these types, assembling systems from CCM components, visualizing various dependence relationships between components, specifying and verifying correctness properties of models of CCM systems derived from CCM IDL, component assembly information, and Cadena specifications, and producing CORBA stubs and skeletons implemented in Java. Cadena has been applied to build applications in Boeing's Bold Stroke framework for avionics mission-control systems. Cadena is implemented in IBM's Eclipse open-source IDE and is freely available.

As software systems become more distributed, developers are increasingly turning to component-based development frameworks such Java Enterprise Beans (EJB) and the CORBA Component Model (CCM) to manage the complexities associated with building distributed systems. These frameworks aid application developers by providing services for common aspects such as distributed deployment, event notification, transactions, persistence, and security. Moreover, they use accepted design patterns (e.g., the event-oriented observer pattern) which enables a significant amount of code to be auto-generated. Component-based frameworks are also attractive because the relatively loose coupling between components facilitates reuse and allows systems to evolve gracefully as old components are switched out for new ones.

Even in the domain of distributed real-time embedded (DRE) systems where hard/soft deadlines and minimal foot-print requirements traditionally have led

* This work was supported in part by the U.S. Army Research Office (DAAD190110564), by DARPA/IXO's PCES program (AFRL Contract F33615-00-C-3044), by Lockheed-Martin, Rockwell-Collins and by Intel Corporation (Grant 11462)

developers to eschew sophisticated middleware solutions, component-based infrastructures are growing more popular because hardware advances allow real-time and embedded requirements to be more easily achieved. In addition, component-based infrastructures provide a framework for systematically introducing important domain aspects such time-triggered notification, real-time scheduling, and fault tolerance.

There is a wide body of literature dealing with the theory of modeling distributed systems and automated analysis of high-level state-based models using state-space exploration techniques such as model-checking. However, despite the popularity of component-based frameworks and their potential to be utilized in mission- and safety-critical applications, relatively little has been done to scale up these analysis techniques for the purpose of providing automated analysis tools for component frameworks. This is particularly the case with CCM – partly due to the fact that the CCM specification as part of CORBA 3.0 has only recently been finalized. Popular tools such as Rational Rose do not even provide design support for CCM yet.

To investigate the effectiveness of a variety of behavioral analysis techniques for component-based systems, we have built *Cadena* [1] – an integrated development environment for high-assurance CCM-based systems.

Cadena provides the following fully implemented capabilities for development of CCM systems.

- A collection of light-weight specification forms that can be attached to IDL to specify mode variable domains, intra-component dependencies, and component state-transition semantics. These forms have a natural refinement order so that useful feedback can be obtained with little annotation effort, and increasing the precision of annotation yields more precise analysis. In addition, Cadena specifications allow developers to specify the same information in different ways, achieving a form of *checkable redundancy* that is useful for exposing design flaws.
- Dependency analysis capabilities that allow tracing inter/intra-component event and data dependencies, as well as algorithms for synthesizing dependency-based real-time and distribution aspect information.
- A novel model-checking infrastructure (based on our Bogor model-checking framework [4]) dedicated to event-based inter-component communication via real-time middleware enables system design models (derived from CCM IDL, component-assembly descriptions and annotations) to be model-checked for global system properties.
- Java component stub and skeleton code generated using the OpenCCM [2] CCM IDL to Java compiler.
- A component assembly framework supporting a variety of visualization and programming tools for developing component connections.

[1] "Cadena" is a Spanish word meaning "chain" or "network". Cadena is also an acronym for Component Architecture Development ENvironment for Avionics systems.

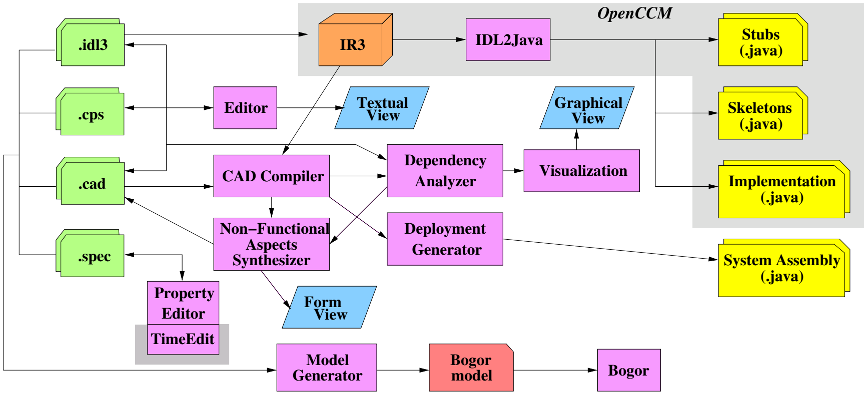

Fig. 1. Cadena architecture

- A CCM deployment facility dedicated to the Boeing Bold Stroke architecture (static component connections with a real-time event-channel) that allows deployment code to be automatically generated.
- The Cadena tools are implemented as plug-ins to IBM's Eclipse IDE. This provides an end-to-end integrated development environment for CCM-based Java systems.

Figure 1 displays the internal structure of the Cadena toolset. Cadena projects contain four high-level specification forms: a CORBA 3 IDL file that defines the structure of component types, a Cadena Property Specification (CPS) file that specifies various aspects of component behavior including abstract state transition semantics and information that specifies dependences between component ports, a Cadena Assembly Description (CAD) that specifies the components instances that form the system, the connections between them, along with other real-time and distribution property information, and a specification file that stores information about the desired correctness properties of the system. These input artifacts are created using customized editors built using Eclipse plug-in facilities. In particular, the CAD format has a textual editor, a graphical editor, and a form-based editor that allows one to easily define different projections of the component assembly (e.g., connections only, particular component attributes only, etc.). The graph structure described by the CAD is the basic data structure that is used by the dependency analyzer, the graphical view displayer, and the deployment code generator (which generates Java code to allocate and connect components).

Cadena uses the OpenCCM tools [2] to generate system implementations. The OMG CORBA 3.0 specification standardizes a strategy for compiling IDL (of which the CCM IDL is part) down to CORBA IDL 2, which can then be translated to an underlying implementation language such as Java or C++. This translation process automatically generates a substantial amount of infrastructure code for tasks such as component creation and connecting and disconnecting ports. The output code contains the usual CORBA *stubs* and *skeletons*, along with skeleton *implementations* of component methods and event handlers. With this code generation, the developer only needs to implement event handlers and methods on provided interfaces. In future work we are exploring the extension of CCM-based code generation strategies to integrate code generation for component handler state-machines and global synchronization policies.

When building systems with Cadena, we intend for developers to take the following steps: (1) load a library of domain-specific components and associated CPS specifications, (2) define new project-specific components and associated behavioral CPS specifications, (3) use CAD editors to configure connections between components, (4) use dependency viewer to examine dependencies, (5) use non-functional aspect synthesis tools to attach distribution and rate information, (6) specify desired global correctness properties, (7) generate a transition system model and model-check correctness properties, and (8) revise system based on feedback from analysis tools.

Up to this point, Cadena has been applied primarily to develop representative applications from Boeing's Bold Stroke avionics mission control software framwork. We have worked with engineers from both Boeing and Rockwell-Collins with a goal to design Cadena and its associated use methodology so that it could be integrated into the actual Bold Stroke development process. In fact, we believe that this "customer-driven" context is one of the things that makes this work interesting and relevant: we address analysis of widely-used general purpose middleware frameworks and languages, and we design the functionality and features of our analysis tools to mesh with an actual industrial development process.

Although Cadena was originally targeted to the avionics domain, it is useful in many respects for CCM development in general. Even though it currently emphasizes Java in its back-end facilities, since CCM is language-neutral, Cadena's front-end design capabilities are not Java dependent. For example, we are also working closely with researchers developing CIAO [1] (a C++ CCM implementation based on the ACE/TAO real-time middleware framework) to integrate CIAO into Cadena, and to refine the Cadena APIs to support specification and modeling of real-time and quality-of-service properties.

There are other development systems that support several important aspects for DRE systems that Cadena does not, such as timing and schedulability analysis, reliability and fault analysis, as well as sophisticated deployment strategies. The primary motivation for our work is to build a system that is robust enough for development of real systems with the goal of assessing the effectiveness of applying static analysis, model-checking, and other light-weight formal methods to CCM-based systems.

The technical foundations of Cadena were presented in [3]. More information about Cadena including a public distribution, papers, tutorials, talks, example repository, and guidelines for other researchers wishing to integrate their analysis tools or CCM implementations into Cadena can be found at the Cadena web site [5].

References

1. DOC Group – CIAO Development Team. CIAO Website. `http://www.cs.wustl.edu/~nanbor/projects/CIAO`, 2003.
2. GOAL. The OpenCCM platform. `http://corbaweb.lifl.fr/OpenCCM/`, 2002.
3. J. Hatcliff, W. Deng, M. Dwyer, G. Jung, and V. Prasad. Cadena: An integrated development, analysis, and verification environment for component-based systems. In *Proceedings of the 25th International Conference on Software Engineering (to appear)*, 2003.
4. Robby, M. B. Dwyer, and J. Hatcliff. Bogor Website. `http://bogor.projects.cis.ksu.edu`, 2003.
5. SAnToS Laboratory. Cadena Website. `http://cadena.projects.cis.ksu.edu`, 2003.