MDA-based Configuration of Distributed Real-time and Embedded Systems

Brian Dougherty, Jules White, and Douglas C. Schmidt
Institute for Software Integrated Systems
Vanderbilt University, Nashville, TN 37203, USA
{ briand, jules, schmidt }@dre.vanderbilt.edu

Section 1. Introduction

Emerging trends and challenges. Distributed real-time embedded (DRE) systems (such as avionics systems, satellite imaging systems, smart cars, and intelligent transportation systems) are subject to stringent requirements and constraints. For example, timing constraints require that tasks be completed by real-time deadlines. Likewise, rigorous quality of service (QoS) demands (such as dependability and security), may require a system to recover and remain active in the face of multiple failures [Wang 2003]. In addition, domain-specific constraints (such as the need for power management in embedded systems) must be satisfied. To cope with these complex issues, applications for DRE systems have traditionally been built from scratch using specialized, project-specific software components that are tightly coupled with specialized hardware components [Schmidt 2002].

To reduce development cycle-time and cost, however, the new generation of DRE systems is increasingly being developed by *configuring* applications from multiple layers of commercial-off-the-shelf (COTS) hardware, operating systems and middleware components [Voas 1998]. These types of DRE systems require the integration of 100's-1,000's of software components that provide distinct functionality (such as I/O, data manipulation, and data transfer) that must work in concert with other software to accomplish mission-critical tasks (such as self-stabilization, error notification, and power management). The software configuration of a DRE system directly impacts its performance, cost, and quality.

As COTS-based DRE systems increase in size and complexity, however, traditional design techniques based on complete in-house proprietary construction are not sufficient to configure COTS-based DRE systems that can simultaneously address the stringent requirements and constraints outlined above [Gokhale 2002]. The objective of DRE system configuration is therefore to determine exactly what combination of hardware/software components will provide the requisite QoS. In addition, the combined purchase cost of the components cannot exceed a predefined amount, referred to as the *project budget*.

An overall DRE system configuration can be split into a software configuration and a hardware configuration. A valid software configuration must meet all real-time constraints (such as minimum latency and maximum throughput) provide required functionality, and also satisfy all domain-specific design constraints while not exceeding the available budget for purchasing software components. Similarly, the hardware configuration must meet all constraints without exceeding the available hardware component budget.

It is likely that for each portion of desired software functionality, there are multiple COTS components that can perform the desired function. Each option differs in QoS provided, the amounts/types of computational resources required, and purchase cost. The complexity associated with DRE systems composed of 100's-1,000's of components makes it hard to find configurations that meet complex QoS constraints. Creating and maintaining error-free configurations is also hard due to the large number of complex configuration rules and QoS requirements.

Solution approach → **Model-driven automated configuration techniques**. This paper presents techniques and tools that leverage the *Model Driven Architecture* (MDA) paradigm to determine valid DRE system configurations that fit budget limits. To help address the difficulty of DRE system configuration, DRE system designers can use MDA to visualize configuration options/rules, verify configuration validity, and compare potential DRE system configurations.

MDA is a design approach for specifying system configuration constraints with platform-independent models (PIMs). Each PIM can be used as a metamodel for constructing platform-specific models (PSM)s [Poole 2001]. These PSMs can be analyzed to determine DRE system configurations that meet budget constraints. After a PSM is determined for implementation, it can be used as a blue print for constructing an actual DRE system implementation that meets all design constraints specified within the PIM [Kent 2002]. As DRE system requirements evolve and additional constraints are introduced, the PIM can be modified and new PSMs constructed. Systems that are constructed using these PSMs can reflect additional constraints and requirements more readily than those developed manually using third-generation languages.

This chapter describes and evaluates MDA-based analyses techniques for determining high quality DRE system configurations. We survey metamodeling techniques for creating DSMLs that can be applied to DRE system

configuration and demonstrate the creation of a DSML for modeling hardware/software component options, resource constraints, and budgetary constraints. We also show how to utilize modeling environments to create models that adhere to the DSML for DRE system configuration. In addition, we demonstrate an interpreter that can examine models of hardware/software DRE system configuration options, generate code, which can ultimately be used to produce output models that provide valid, high-quality large-scale DRE system configurations.

Section 2: Large-scale DRE System Configuration Challenges. This section presents the criteria for valid DRE system configurations, describes the challenges that make determining configurations hard, and provides a survey of current techniques and methodologies for DRE system configuration. Software and hardware components often have complex interdependencies on the consumption and production of resources (such as processor utilization, memory usage, and power consumption). An overall DRE system configuration consists of a valid hardware configuration and valid software configuration in which the computational resource needs of the software configuration are provided by the computational resources produced by the hardware configuration. If the resource requirements of the software configuration exceed the resource production of the hardware configuration, a DRE system will not function correctly and is considered invalid.

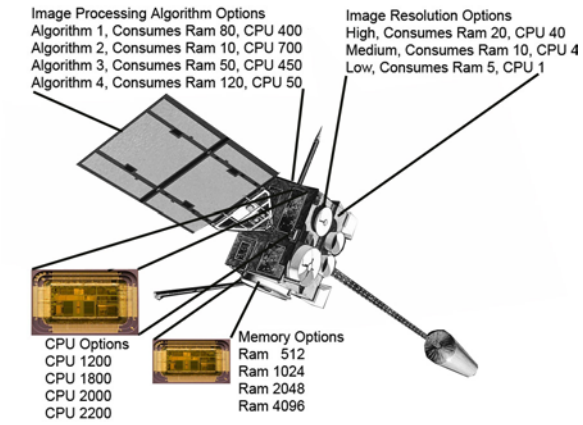

Figure 1 Configuration Options of a Satellite Imaging System

Figure 1 shows the configuration options of a satellite imaging system. This DRE system consists of two software components: an image processing algorithm and software that defines image resolution capabilities. There are multiple options for each software component, each of which provides a different level of service. For example, there are three options for the image resolution component. The high-resolution option offers the highest level of service, but also requires dramatically more RAM and CPU to function than the medium or low-resolution options. If RAM or CPU resources are scarce, therefore, the medium or low-resolution option should be chosen. An additional design constraint, however, may require at least medium image resolution. Assuming sufficient resources, the only option that satisfies all constraints is the medium image resolution option.

Large-scale DRE systems may consist of many software and hardware components with multiple options for each component, resulting in an exponential number of potential configurations. The huge magnitude of the solution space prohibits the use of manual techniques. Automated techniques, such as Constrained Linear Programming (CLP), use Constraint Satisfaction Problems (CSPs) to represent system configuration problems [Benavides 2003, Sabin 1996]. These techniques are capable of determining optimal solutions for small-scale system configurations but require the examination of all potential system configurations.

The exhaustive nature of conventional CSP-based techniques, however, renders them ineffective for large-scale DRE system configuration. Without tools to aid in large-scale DRE system configuration, it is a struggle for designers to determine *any* valid large-scale system configuration. Even if a valid configuration is determined, other valid system configurations may exist with vastly superior performance and dramatically less financial cost. It is thus imperative that advanced design techniques, such as MDA, are developed to enhance and validate large-scale DRE system configurations.

The following is a summary of key challenges that make it particularly hard to configure DRE systems using COTS components:

- **Choosing between multiple levels of service.** Software components provide differing levels of service. For example, a designer may have to choose between three different software components that differ in speed and

throughput. In some cases, a specific level of service may be required, prohibiting the use of certain components.

- **Complex resource interdependencies.** Hardware components provide the computational resources that software components require to function. If the hardware does not provide and adequate amount of each computational resource, some software components cannot function. An overabundance of resources indicates that some hardware components have been purchased unnecessarily, wasting funds that could have been spent to buy superior software components or set aside for future projects.
- **Satisfying differing resource requirements.** Each software component requires computational resource to function. These resource requirements differ between components. Often, components offering higher levels of service require larger amounts of resources and/or cost more to purchase. Designers must therefore consider the additional resulting resource requirements when determining if a component can be included in a system configuration.
- **Meeting budgetary constraints.** Each component has an associated purchase cost. The combined purchase cost of the components included in the configuration must not exceed the project budget. It is therefore possible for the inclusion of a component to invalidate the configuration if its additional purchase cost exceeds the project budget regardless of computational resources existing to support the component.
- **Choosing from many components.** Large-scale DRE systems require hundreds of components to function. Each component may be many options available for inclusion in the final system configuration. Due to the complex resource interdependencies, budgetary constraints, and domain- specific design constraints it is hard to determine if including a single component will invalidate the system configuration. This problem is exacerbated enormously if designers are faced with the tasks of choosing from 1,000's of available components. Even automated techniques require years or more to examine all possible system configurations for such problems.

Subsequent sections of the chapter will demonstrate how MDA can be utilized to mitigate many of the difficulties of DRE system configuration that are a result of the aforementioned challenges.

Section 3: Background of System Configuration Optimization. This section surveys various DRE system configuration problems and model analysis techniques, including hardware/software co-configuration problems and heuristic algorithms for resource-constrained configuration. We will describe several different types of DRE system configuration problems. We will also examine several techniques that have been applied to aid in the determination of optimal or near-optimal DRE system configurations, such as Constrained Linear Programming and Constraint Satisfaction Problems. Finally, we will describe how MDA can be used to mitigate many of the problems associated with these techniques.

Section 4: MDA-based Configuration Modeling Methods and Tools. This section describes metamodeling and modeling techniques in the domain of DRE system configuration. Due to the complexities accompanying system configuration with COTS components (such as differing levels of service, complex resource interdependencies, differing resource requirements, budgetary constraints, and a multitude of component candidates), designers using manual techniques often unknowingly invalidate system configurations. MDA tools allow designers to manipulate problem entities and compare potential solutions in an environment that ensures various design rules are enforced, thereby allowing designers to focus on other problem dimensions, such as performance optimization or minimization of computational resources.

The Domain Specific Modeling Languages (DSMLs) defined by PIMs describe the rules and constraints for constructing a model of a problem arising within a particular domain. We will describe the process of utilizing metamodeling environments to construct PIMs that define a DSML for DRE system configuration problems. We will present a procedure for applying this DSML to create valid system configuration models. We will also present a modeling environment for creating PSMs that reflect the design constraints of the DSML. Finally, we will introduce an interpreter, a tool that examines PSMs and generates code described by the PSM. The examination of existing modeling methods and tools is essential for understanding and constructing new modeling techniques.

Section 5: Case Study. This section will describe our MDA-based approach for modeling hardware/software system configuration and determining near-optimal, valid configurations on a DRE system software configuration case study. We will show readers how to create a PIM defining a DSML for DRE system configuration using MetaGME, a metamodeling environment for constructing DSMLs. Metamodels created with MetaGME can be used in conjunction with GME (Generic Modeling Environment) to create models that can be interpreted using a BON2 interpreter. We will describe how the BON2 interpreter can generate Java code, allowing analysis of the system

configuration model. Finally, we will show the BON2 interpreter can be utilized to automatically reflect results in a GME model.

Section 6: Concluding Remarks. This section will summarize lessons learned from the design, implementation, and execution of the MDA-based configuration modeling methods and tools described in Section 4, including the potential extensions. For example, co-location constraints of software components are not currently modeled in the PIM. Many DRE system configurations require or disallow the placement of certain components on the same hardware component. It would therefore be necessary to extend the PIM so it is applicable for additional system configuration problems involving collocation constraints.

References

1. D. Benavides, P. Trinidad, and A. Ruiz-Cortes. Automated Reasoning on Feature Models. *17th Conference on Advanced Information Systems Engineering (CAiSE05, Proceedings), LNCS*, 3520:491–503, 2003.
2. A. Gokhale, D. Schmidt, B. Natarajan, and N. Wang, "Applying Model-Integrated Computing to Component Middleware and Enterprise Applications," *Communications of the ACM*, 45(10):65–70, 2002.
3. S. Kent, "Model Driven Engineering," *In Proceedings of Third International on Integrated Formal Methods*, pages 286–298. Springer-Verlag London, UK, 2002.
4. J. Poole, "Model-driven architecture: Vision, Standards and Emerging Technologies," In *Workshop on Meta-modeling and Adaptive Object Models, ECOOP*, 2001.
5. D. Sabin and E. Freuder, "Configuration as Composite Constraint Satisfaction," in *Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop*, pages 153–161. AAAI Press, 1996.
6. D. Schmidt, "Middleware for Real-time and Embedded Systems," *Communications of the ACM*, 45(6):43–48, 2002.
7. J. Voas, "Certifying Off-the-shelf Software Components," *IEEE Computer*, 31(6):53–59, 1998.
8. N. Wang, D. Schmidt, A. Gokhale, C. Gill, B. Natarajan, C. Rodrigues, J. Loyall, and R. Schantz, "Total Quality of Service Provisioning in Middleware and Applications," *Microprocessors and Microsystems*, 27(2):45–54, 2003.

Relevance for this Book

The use of Model Driven Architectures to facilitate design processes is an important theme of this book. This chapter presents a description of DRE system configuration using COTS components and describes the complexities associated with it. We demonstrate how Model Driven Architectures can facilitate the design process, greatly mitigating these associated difficulties. Finally, we present a case study that proves the effectiveness of applying MDA to this technique, thereby motivating the use and continued study of MDA.

Author Biographies

Brian Dougherty is a Ph.D candidate in Computer Science at Vanderbilt University. Brian's research focuses on hardware/software co-design, heuristic constraint-based deployment algorithms, and design space exploration. He is the co-leader of the ASCENT project, a tool for analyzing hardware/software co-design solution spaces. Brian is also a developer for the Generic Eclipse Modeling System (GEMS). He received his B.S. in Computer Science from Centre College, Danville, KY in 2007.

Dr. Jules White is a Research Assistant Professor at Vanderbilt University. He received his BA in Computer Science from Brown University, his MS in Computer Science from Vanderbilt University, and his Ph.D. in Computer Science from Vanderbilt University. Dr. White's research focuses on applying a combination of model-driven engineering and constraint-based optimization techniques to the deployment and configuration of complex software systems. Dr. White is the project leader for the Generic Eclipse Modeling System (GEMS), an Eclipse Foundation project.

Dr. Douglas C. Schmidt is a Professor of Computer Science and Associate Chair of the Computer Science and Engineering program at Vanderbilt University. He has published 9 books and over 400 papers that cover a range of topics, including patterns, optimization techniques, and empirical analyses of software frameworks and domain-specific modeling environments that facilitate the development of distributed real-time and embedded (DRE) middleware and applications. Dr. Schmidt has over fifteen years of experience leading the development of ACE, TAO, CIAO, and CoSMIC, which are open-source middleware frameworks and model-driven tools that implement patterns and product-line architectures for high-performance DRE systems.