Model-Driven Generation of DSML Execution Engines*

Gustavo C. M. Sousa
Fábio M. Costa
Instituto de Informática
Universidade Federal de Goiás
Goiânia-GO, Brazil
{gustavo|fmc}@inf.ufg.br

Peter J. Clarke
School of Computing and
Information Sciences
Florida International University
Miami-FL, USA
clarkep@cis.fiu.edu

ABSTRACT

The combination of domain-specific modeling languages and model-driven engineering techniques hold the promise of a breakthrough in the way applications are developed. By raising the level of abstraction and specializing in building blocks that are familiar in a particular domain, it has the potential to turn domain experts into application developers. Applications are developed as models, which in turn are interpreted at runtime by a specialized execution engine in order to produce the intended behavior. This approach has been successfully applied in different domains, such as communication and smart grid management. However, each time the approach has to be realized in a different domain, substantial re-implementation has to take place in order to put together an execution engine for the respective DSML. In this paper, we present our work towards a generalization of the approach in the form of a meta-model and its respective execution environment, which capture the domain-independent aspects of runtime model interpretation and allow the definition of domain-specific execution engines as instances of the meta-model. We present an initial validation of the approach in the context of the Communication Virtual Machine project, by realizing part of the execution engine architecture in the form of an instance of the proposed meta-model.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*domain-specific architectures, languages*

Keywords

Models at Runtime, Model-Driven Engineering, Domain-Specific Modeling Languages, Metamodeling, Middleware

1. INTRODUCTION

*This work was partly supported by the Capes Foundation, Brazil, Proc. 0759-11-2

Models@Run.Time '12 Innsbruck, Austria

Model driven engineering is being proposed as an approach to deal with the growing complexity involved in the development of modern applications. These applications commonly operate in a distributed and heterogeneous environment and must behave in a dependable manner. These applications also often need to adapt at runtime to comply with changing requirements and environment. In order to achieve these goals the developers need to employ a variety of strategies to

MDE proposes to simplify the development of applications. By employing models built upon abstractions of the problem domain. MDE looks forward reducing the complexity due to the the technical details in the translation from domain concepts to platform concepts.

In order to reduce this complexity, MDE relies on DSMLs and automated processing of models. Models can be either transformed or interpreted.

The use of high level (and maybe graphical) DSMLs? Making it possible for domain experts, or even end users to create applications from their requirements.

Communication Virtual Machine (CVM) and MicroGrid Virtual Machine (MGridVM) are examples of software that employ MDE techniques to allow users to develop applications using DSMLs. Don't forget to mention that its not only applications, but complex applications Besides allowing the development of applications, these software also enables the users to adapt applications during their runtime by manipulating models. These virtual machines are execution engines for DSMLs named CML and MGridML respectively.

to allow the above (user built applications)? DSMLs, along with the mechanisms for their processing embed concepts and semantics of the domain. Once we have a DSML and the mechanism for processing it, users can directly develop applications in that DSML.

While the development of applications in these domains employing their respective DSMLs is quite simple, the same is not true when talking about the execution engine that interprets their languages. These mechanisms are commonly developed in general purpose language and are subject to all the technical complexities involved in the traditional software development.

Despite that, these execution engines share a lot of commonalities. Their applications basically provide a high-level service upon a set of heterogeneous resources. In order to do that, the runtime engine may need to execute many operations related to the processing of models, negotiation of configurations, synthesis of commands, evaluation of policies,

runtime adaptation, and etc. Both CVM and MGridVM are built upon a layered architecture, in which each layer has well defined responsibilities.

Taking that into consideration, we propose a model-driven approach to the construction of execution engines for the provision of high-level services described by DSMLs upon a set of resources. Doing so, we look forward a way to assist the construction of DSML execution engines. Our approach relies on the experience acquired in the development of CVM and MGridVM to propose a generic architecture and the use of modeling as a way to specialize it. We illustrate this approach by presenting a meta-model designed to describe one of the layers of the proposed architecture.

2. BACKGROUND

models@runtime são uma forma de autorepresentação com causalidade i.e. reflexão mais próxima do domínio

3. GENERIC ARCHITECTURE OF THE EXECUTION ENGINE

Dada uma DSML previamente definida, podemos definir um execution engine que vai executar instancias dessa DSML Futuras mudanças no execution engine podem ser para atender mudanças na linguagem ou para mudar requisitos não funcionais ligação entre linguagem e plataforma não é formalizado

4. META-MODEL FOR BROKER LAYER

4.1 Interface

4.2 Signal handling

4.3 Resource management

4.4 State management

4.5 Autonomic computing

4.6 Policies

5. EXECUTION ENVIRONMENT

6. EXAMPLE IN THE COMMUNICATION DOMAIN

7. RELATED WORK

Quais as áreas relacionadas, e trabalhos relacionados - Outras abordagens de construção de execution engines de DSMLs

8. CONCLUDING REMARKS

9. REFERENCES