

Um Meta-modelo para o Desenvolvimento de Aplicações Distribuídas

Lyrene Fernandes da Silva e Virgínia C. Carneiro de Paula
Universidade Federal do Rio Grande do Norte (UFRN)
Departamento de Informática e Matemática Aplicada (DIMAp)
{lyrene, vccpaula}@ufrnet.br

Palavras-chave: Arquitetura de Software, Estilos, Arquitetura em Camadas, UML, CORBA.

Resumo

A crescente complexidade dos sistemas de software tem levado à comunidade de pesquisadores e projetistas a criar ou melhorar técnicas e métodos para o desenvolvimento de software, de maneira a aumentar a satisfação dos clientes e a produtividade dos desenvolvedores. A reutilização de código já não é suficiente para a grande demanda por software de alta qualidade nem para a engenharia deles. Desenvolver componentes de alta abstração é necessário. Porém, reutilizar ou integrar estes componentes a outros são tarefas que requerem um certo trabalho. A escolha de uma boa arquitetura é o primeiro ponto a se pensar. A arquitetura guia o projetista durante o design de um sistema e o ajuda a, antecipadamente, tomar decisões de alto nível e de baixo nível que afetam o comportamento global do sistema. O presente trabalho tem o intuito de apresentar um Meta-modelo para o desenvolvimento de aplicações distribuídas. Este Meta-modelo irá guiar os projetistas na definição de arquiteturas de software consistentes com CORBA (Common Object Request Broker Architecture). Ele é fruto da integração de tecnologias que estão sendo largamente utilizadas por desenvolvedores de software – os estilos e visões arquiteturais, arquiteturas em camadas, UML (Unified Modeling Language), e CORBA – e constitui uma iniciativa de otimizar o *design* de aplicações distribuídas.

Abstract

The growing complexity of the software systems has been leading the researchers and designers to create or to improve techniques and methods for the development, in a way to increase the customers satisfaction and the designers productivity. The code reuse is no longer enough for the great demand for high quality software nor for their engineering. Developing components of high abstraction is necessary. However, reusing or integrating these components to others are tasks that demand an accurate work. The choice of a good architecture is the first point for thinking about. The architecture guides the designer during the design of a system and it also aids him to make decisions early of higher level and of lower level that can affect the global behavior of the system. The present work presents a meta-model for the development of distributed applications. This meta-model will be a guideline to designers in the definition of consistent software architectures with CORBA (Common Object Request Broker Architecture). It is fruit of the integration of technologies that are being broadly used by software designers – the architectural styles and views, architectures in layers, UML (Unified Modeling Language), and CORBA – and it constitutes an initiative of optimizing the design of distributed applications.

1 – Introdução

O uso de estilos e famílias arquiteturais facilita o desenvolvimento de sistemas de software por fornecer um modelo a partir do qual eles podem ser construídos. Ainda assim, conduzir o desenvolvimento de um sistema da arquitetura de família até a arquitetura detalhada de um produto específico é uma tarefa árdua, pois é difícil garantir a consistência entre os artefatos produzidos.

Realizar o refinamento da arquitetura de família antes de fazê-lo para um produto específico, embora igualmente difícil, ajuda no refinamento da arquitetura de qualquer produto com características semelhantes e pode ser reutilizado sempre que necessário [1].

O presente trabalho apresenta um Meta-modelo para o desenvolvimento de aplicações distribuídas. O Meta-modelo define uma arquitetura para uma família de sistemas e provê os blocos de construção básicos para criá-los. Na dissertação, informações de quando e como utilizar este Meta-modelo serão oferecidas, tanto quanto estudos de caso que atestem a sua validade. Este Meta-modelo é fruto da integração do Estilo Camadas, da UML, e de CORBA.

Esta proposta constitui parte da otimização do processo de desenvolvimento de aplicações distribuídas para execução em um middleware. Fornecer as diretrizes para o desenvolvimento destes sistemas é uma tarefa difícil, mas que pode prover benefícios imensuráveis para a engenharia de software visto que, o conhecimento intrínseco a estes sistemas pode ser levantado uma vez e reutilizado inúmeras vezes sem o esforço inicial.

Trabalhos correlatos a este são os trabalhos de Gamma et al. [2], Buschmann et al. [3] e Rosenblum et al. [4]. Os dois primeiros referem-se a proposta de catalogar padrões arquiteturais para que sejam difundidos e facilmente consultados, diferentemente de nosso projeto que não se restringe a estes propósitos. Assim como o nosso trabalho, Rosenblum et al. demonstra preocupação em guiar os projetistas na definição de arquiteturas consistentes com uma infraestrutura de middleware pré-selecionada, entretanto ele utiliza C2 como linguagem enquanto que nós estamos propondo a utilização da UML que é uma linguagem mais difundida.

A seguir, apresentamos como este Meta-modelo está sendo desenvolvido.

2 – A Proposta de Um Meta-Modelo para o Desenvolvimento de Aplicações Distribuídas

O Meta-modelo que estamos propondo é uma solução conceitual para o desenvolvimento de aplicações distribuídas que possam ser estruturadas em camadas e utilizem CORBA como infraestrutura de middleware.

Para elaborar o Meta-modelo utilizamos: o Estilo Camadas [5] como meio de oferecer subsídios para melhor gerenciar a complexidade intrínseca aos sistemas distribuídos; de CORBA [6] para oferecer suporte a interoperabilidade e; da UML [7], para fornecer maior representatividade e comunicabilidade aos modelos ou arquiteturas. Isto facilitará a comunicação entre projetistas e arquitetos e também a transformação de arquiteturas em *design* detalhado do software.

As visões estática e dinâmica representam o Meta-modelo. A visão estática fornece os elementos e as possíveis relações entre eles, além de apresentar o conjunto de restrições imposto pelo estilo camadas e pelo CORBA. A visão dinâmica fornece as interações passíveis de ocorrer entre os elementos e o comportamento interno deles através dos estados que eles podem assumir e da sequência de atividades que eles podem realizar.

A estrutura estática do Meta-modelo teve como ponto de partida a Visão Conceitual, definida por Hofmeister et al. [8], que descreve os elementos (configuração, componentes, conectores e protocolo) de uma arquitetura genérica. Para elaboração deste Meta-modelo subdividimos os problemas em três etapas principais (Figura 1):

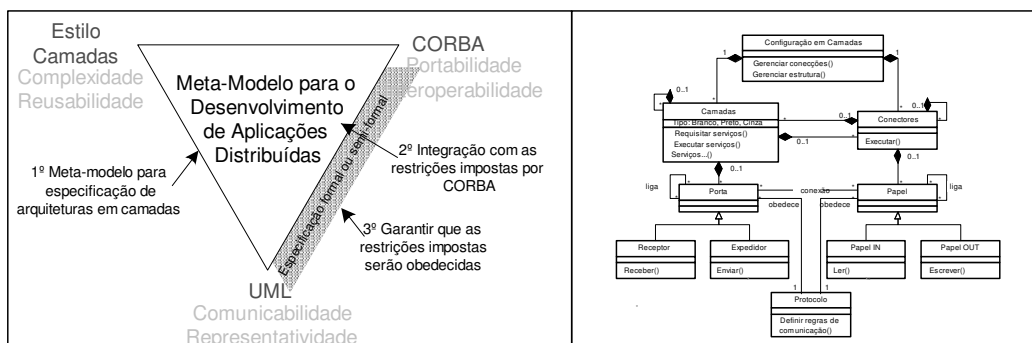


Figura 1: Visão geral das tecnologias envolvidas no meta-modelo proposto

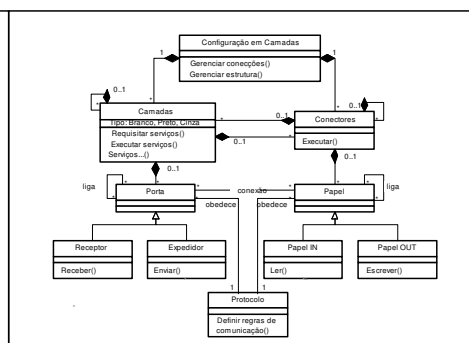


Figura 2: Estrutura estática do Meta-modelo

1º etapa: Elaborar um Meta-modelo para especificação de arquiteturas de software em camadas. A proposta desta etapa é especificar o estilo camadas de maneira que ele possa ser facilmente entendido e utilizado. Para tanto, reunimos várias descrições textuais sobre estilos arquiteturais, em especial, sobre o estilo camadas [3][5][8][9], e propomos um Meta-modelo, especificado em UML [10], para a descrição de arquiteturas de software em camadas. Este trabalho constitui o primeiro passo para que a UML dê suporte à descrição de estilos.

Este Meta-modelo está em fase de conclusão. Um resumo de sua estrutura estática é apresentado na Figura 2. As restrições impostas pelo Estilo Camadas foram especificadas através da linguagem Z. Definiremos ainda, regras para sua utilização e estereótipos para facilitar, ainda mais, a modelagem, o entendimento e a comunicabilidade. O próximo passo a ser cumprido é integrar às restrições impostas por CORBA a este Meta-modelo.

2º etapa: Integrar o Meta-modelo, da etapa anterior, às restrições impostas por CORBA. Para que uma aplicação baseada em CORBA possa efetivamente interoperar com outras é necessário que ela esteja adequada ao modelo proposto por CORBA. A princípio é necessário: obedecer ao modelo cliente-servidor dinâmico; especificar as interfaces dos objetos; fazer o registro dos servidores junto ao ORB e; definir como os objetos acessam os serviços que desejam.

Para efetivação desta etapa do trabalho refinaremos o modelo anteriormente definido para que ele dê suporte as restrições impostas por CORBA. Estamos decidindo se realizaremos este refinamento a partir do modelo anteriormente definido ou a partir dele e do padrão Broker, acrescentando algumas particularidades do CORBA.

Para utilizar o padrão Broker devemos especifica-lo como fizemos para o Estilo Camadas. Assim, disponibilizaremos um Meta-modelo para especificação de arquiteturas baseadas no padrão Broker. Estes dois Meta-modelos significam por si só parte da otimização do *design* de arquiteturas de software, já que eles retratam as restrições, características, operações e estrutura básica destas arquiteturas. Em seguida, temos que garantir que o Meta-modelo resultante obedece e não-ambiguamente define as restrições pretendidas.

3º etapa: Garantir que todas as restrições impostas pelo estilo camadas e por CORBA sejam cumpridas. A especificação das restrições do Meta-modelo resultante em uma linguagem formal ou semi-formal irá garantir a completude e corretude dele. A decisão de que linguagem utilizar para este propósito será escolhida de acordo com as exigências do modelo final. Após realização desta etapa definiremos quando e como utilizar o Meta-modelo proposto e o aplicaremos a estudos de caso para demonstrar o seu uso.

3 - Contribuições Esperadas e Trabalhos Futuros

Este trabalho une tecnologias amplamente utilizadas no desenvolvimento de sistemas de software, tais como os estilos, padrões e visões arquiteturais, UML, especificação formal ou semi-formal e CORBA. Propomos um Meta-modelo para auxiliar no desenvolvimento de aplicações distribuídas. Este Meta-modelo tem o intuito de dar suporte à etapa de *design* da arquitetura de um sistema distribuído e heterogêneo por oferecer todas as informações necessárias em uma linguagem de fácil entendimento e popular.

Além disto, a conclusão deste trabalho oferecerá, à comunidade de desenvolvedores, outras vantagens como: a primeira etapa para que a UML suporte estilos; suporte formal ou semi-formal à UML; um Meta-modelo para especificação de sistemas de software em camadas e, também, baseados no padrão Broker; maior facilidade de comunicação entre arquitetos e projetistas e; maior facilidade para reusar, entender e manter os sistemas ou subsistemas desenvolvidos.

Esta união significa parte da automação da utilização de estilos no desenvolvimento de sistemas pois, com o uso da UML transferem-se todos os benefícios desta linguagem para o uso de padrões arquiteturais. Por exemplo, ferramentas CASE, ferramentas e técnicas de análise, integração dos modelos de *design* às linguagens de programação e linguagens de descrição de arquiteturas, dentre outros.

Outros trabalhos promissores podem ser realizados nesta mesma linha de estudo, por exemplo: descrever outros estilos como o definido aqui; definir uma metodologia para a utilização deles tanto na engenharia quanto na reengenharia; aplicá-los e testá-los em vários contextos; definir arquiteturas de referência e arquiteturas de linha de produto e; desenvolver uma ferramenta de suporte ao uso destes Meta-modelos e estilos. Estas são tarefas indispensáveis para se obter uma biblioteca de estilos e arquiteturas inteiramente modelados e prontos para serem instanciados.

4 - Referências Bibliográficas

- [1] EGYED Alexander, MEHTA Nikunj e MEDVIDOVIC Nenad: Software Connectors and Refinement in Family Architectures, 3º International Workshop – Software Architectures for Product Families, Las Palmas de Gran Canaria, Spain, March 15-17, 2000.
- [2] GAMMA Erich, HELM Richard, JOHNSON Ralph e VLISSIDES John: Design Patterns – Elements of Reusable Object-Oriented Software, Addison Wesley, 1995.
- [3] BUSCHMANN Frank, MEUNIER Regine, ROHNERT Hans, SOMMERLAD Peter, STAL Michael: A system of patterns-Pattern Oriented Software Architecture, Wiley, 1996.
- [4] ROSENBLUM David e NITTO Elisabetta Di: Exploiting ADLs to Specify Architectural Styles Induced by Middleware Infrastructures, 21st International Conference on Software Engineering (ICSE'99), Los Angeles, CA, pp. 13-22, 1999.
- [5] JACOBSON Ivar, GRISS Martin e JONSSON Patrick: Software Reuse, Addison Wesley, 1997.
- [6] OMG: The object model, www.omg.org
- [7] BOOCH Grady, RUMBAUGH James e JACOBSON Ivar: The Unified Modeling Language User Guide, Addison Wesley, 1999.
- [8] HOFMEISTER Christine, NORD Robert, SONI Dilip: Applied Software Architecture, Addison Wesley, 2000.
- [9] SHAW Mary, GARLAN David: Software Architecture – Perspectives on an Emerging Discipline, Prentice-Hall, 1996.
- [10] SILVA Lyrene F. e PAULA Virgínia C.C. de: Um Meta-Modelo para Especificação de Arquiteturas de Software em Camadas, submetido e aceito para o XV Simpósio Brasileiro em Engenharia de Software (SBES2001), que se realizará em 03-05/10/2001.