Replicação Orientada a Metaprogramação

Fellipe Augusto Ugliara[1]
Orientador: **Prof. Dr. José de Oliveira Guimarães**[1]
Coorientador: **Prof. Dr. Gustavo Maciel Dias Vieira**[1]

[1]Universidade Federal de São Carlos (UFSCar)
Sorocaba – SP – Brasil
Programa de Pós-Graduação em Ciência da Computação – PPGCC-So

Resumo. *Replicar dados é uma técnica muito poderosa usada pela computação distribuída. O conceito de metaprogramação pode ser aplicado em diferentes contextos, neste trabalho ele está relacionado a compilação das aplicações que pretendemos replicar. A metaprogramação permite que em determinadas etapas de compilação o código-fonte de uma aplicação não replicada seja usado para gerar um código-fonte no qual essa mesma aplicação seja replicável. Outras ações de validação também podem ser realizadas durante a geração desse código-fonte, com a intenção de alertar sobre possíveis problemas existentes nos códigos-fonte originais. Nesse trabalho é discutida uma possível solução de como validar se um código-fonte é determinista, uma característica necessária do código-fonte de uma aplicação replicável. A contribuição desse trabalho é mostrar que a metaprogramação pode auxiliar na implementação de aplicações replicadas, como também instigar abordagens similares que possam ser aplicadas a programação de outros requisitos não funcionais.*

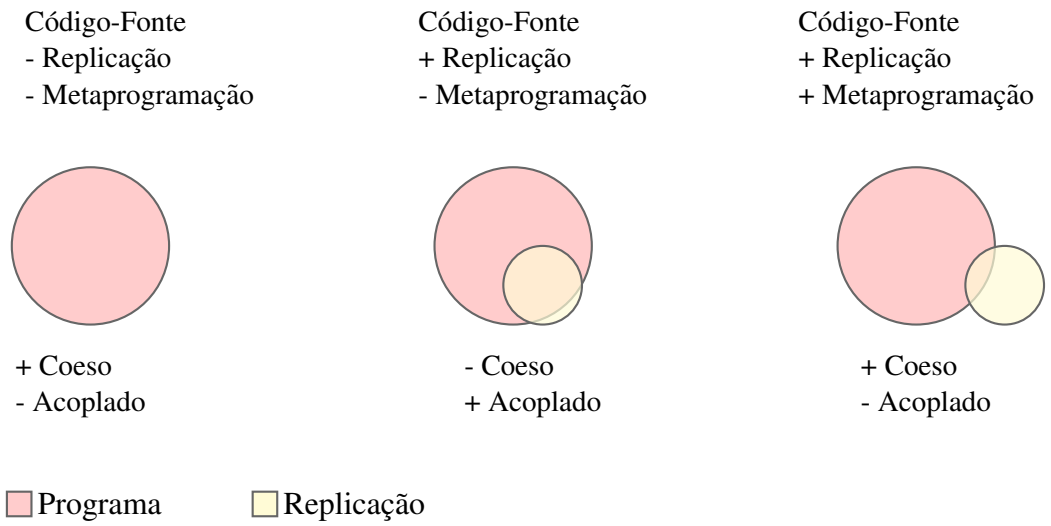
Abstract. *Data replication is a powerful technique of distributed computing. The concept of metaprogramming can be implemented in different contexts; in this work it is related to the compilation of applications we intend to replicate. In certain compilation steps metaprogramming allows the source code of a centralized application to be used to generate a source code in which the same application is replicated. Other validation actions can also be performed during this source code generation in order to warn the user about problems that possibly exist in the original source code. In this work, a possible solution of how to validate if a source code is deterministic is discussed, since it is a necessary characteristic of the source code of a replicable application. This work aims to not only show that metaprogramming can help the implementation of replicated applications, but also instigate similar approaches that can be employed in the programming of other non-functional requirements.*

1. Introdução e Motivação

A implementação de replicação em uma aplicação tem como objetivo melhorar a confiabilidade, tolerância a falhas e acessibilidade dessa aplicação. O requisito de confiabilidade define que ela estará disponível a qualquer momento que um usuário precise utiliza-lá, tolerância a falhas garante que caso a aplicação sofra um mal súbito ela poderá retomar a execução do ponto onde esse problema ocorreu, e acessibilidade define que um usuário poderá acessar a aplicação de qualquer local que for necessário.

A replicação é um requisito não funcional que resolve problemas referentes a tecnologia e não ao domínio para o qual uma aplicação é desenvolvida. Os mesmos requisitos não funcionais pode ser atribuídos a aplicações de domínios diferentes. Uma aplicação financeira pode implementar replicação para atender aos requisitos mencionados, assim como uma aplicação de comercio eletrônico pode usar dessa mesma implementação de replicação para atender a esses mesmos requisitos não funcionais.

Como a replicação não é a motivação primária do desenvolvimento de uma aplicação, o desejo é que a implementação da replicação seja o mais coeso e menos acoplado possível com o restante do código-fonte implementado. A coesão da implementação é uma métrica que define se as partes (módulos, componentes) do código-fonte de uma aplicação tratam cada uma de um único assunto (problema, objetivo, funcionalidade). O acoplamento é uma métrica usada para definir o quanto dependente são essas partes do código-fonte. Para que essas partes de código-fonte sejam reutilizáveis e de fácil manutenção elas devem ser mais coesas e menos acopladas.



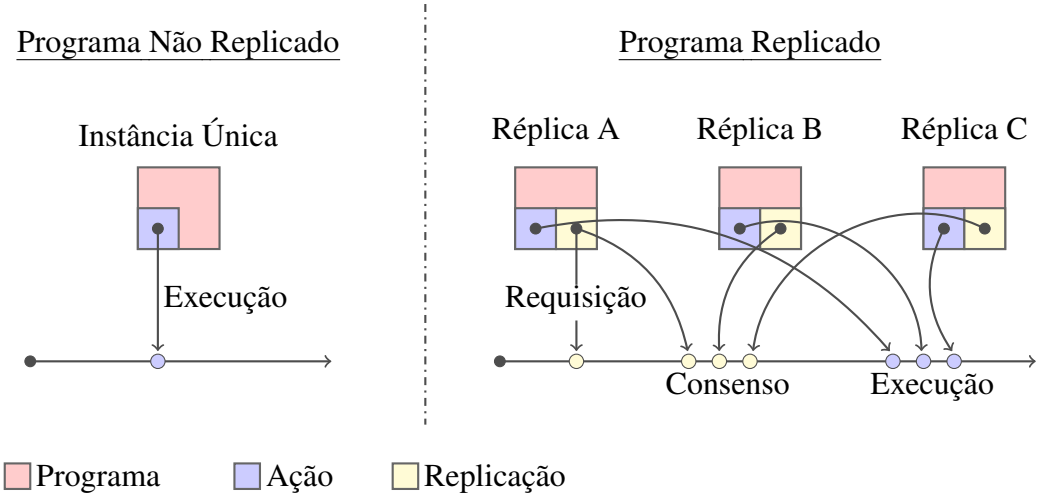
A motivação desse trabalho é mostrar como a metaprogramação em compilação pode auxiliar na implementação de aplicações replicáveis, tornando os códigos-fonte dessas implementações mais coesos e menos acoplados [Ugliara 2018]. O prefixo 'meta' em metaprogramação define uma referencia a si mesmo, como em metadados que são os dados que contêm informação a respeito dos próprios dados. Metaprogramação é o conceito que permite definir programas que referenciam entidades da própria programação (compiladores, interpretadores, códigos-fonte, programas, elementos da linguagem utilizada) [Damaševičius and Štuikys 2015], para definir como esses programas devem ser trabalhados por essas entidades.

Neste trabalho, usando metaprogramação em compilação, foi possível desenvolver programas sem replicação que durante a compilação são usados como entrada para gerar novos programas com replicação [Guimarães 2018b]. Esses novos programas também mantêm as funcionalidades dos programas originais. Na compilação foi possível verificar esses programas na busca de trechos de código-fonte não deterministas que possam levar as aplicações replicadas a apresentarem problemas durante a execução.

O determinismo em aplicações replicadas é necessário porque cada ação gerada em uma réplica dessa aplicação será compartilhada com as demais réplicas que estive-

rem vinculadas a um mesmo grupo de execução. Caso essas ações não sejam deterministas as réplicas poderão assumir comportamentos diferentes o que tornaria a execução desse grupo de réplicas inconsistente [Vieira and Buzato 2010]. Um grupo de execução é um conjunto de réplicas, de uma mesma aplicação, que estão vinculadas a uma mesma execução.

Todas as réplicas de um grupo de execução serão iguais, assim se uma réplica sofrer um mal súbito ela poderá ser recriada a partir de outra réplica que esteja no mesmo grupo, o que melhora a tolerância a falhas das aplicações. As requisições dos usuários de aplicações replicadas podem ser distribuídas entre as réplicas para melhorar a confiabilidade. As réplicas também podem ser executadas em pontos geográficos diferentes para melhorar a acessibilidade dessas aplicações.



Desenvolver aplicações replicadas exige uma certa habilidade por parte da equipe de desenvolvimento para que a aplicação seja determinista, também é necessário que bibliotecas ou *frameworks* que realizam a replicação sejam estudados o que pode exigir tempo e maturidade dessa equipe. Usar metaprogramação nesse cenário melhora o desenvolvimento, pois torna a replicação transparente para a equipe de desenvolvimento, o que reduz a necessidade de compreender como essas bibliotecas ou *frameworks* funcionam. A metaprogramação também melhora o tempo gasto em desenvolvimento que se torna menor uma vez que a responsabilidade de tornar a aplicação replicável fica a cargo da metaprogramação e não da equipe de desenvolvimento.

2. Replicação e Metaprogramação

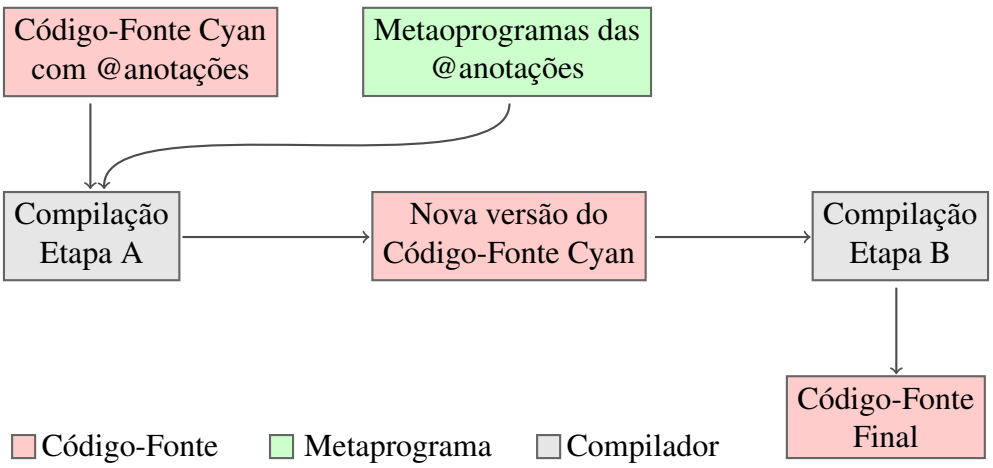
Esse documento é um resumo da dissertação *Replicação Orientada a Metaprogramação* [Ugliara 2018]. Nessa dissertação a abordagem explicada na sessão anterior foi desenvolvida e validada, usando o *framework* Treplica [Vieira and Buzato 2008] para prover replicação e a linguagem de programação Cyan [Guimarães 2018a] por permitir metaprogramação em compilação. Aqui será apresentado um resumo desse trabalho, e na próxima sessão as conclusões e impactos da pesquisa.

Treplica é um *framework* baseado em estados e ações. As aplicações desenvolvidas com ele implementam seu comportamento no formato de ações que quando passados à máquina de estados de Treplica, alteram seu estado interno. Essas ações quando enviadas da aplicação para máquina de estados de Treplica também são repassadas para o

grupo de réplicas a qual a réplica que enviou a ação está vinculada. Essas ações quando recebidas pelas demais réplicas modificam o estado interno da máquina de estados de Treplica de quem a recebeu. Desse modo todas as réplicas de um grupo sempre estarão com a máquina de Treplica no mesmo estado, essas réplicas serão iguais.

A metaprogramação em Cyan é aplicada durante a compilação das aplicações. Nessa linguagem é possível desenvolver programas (metaprogramas) que são executados pelo compilador de Cyan no momento da compilação de um código-fonte. Com base em elementos encontrados no código-fonte que está sendo compilado esses metaprogramas são executados pelo compilador para gerar uma nova versão do código-fonte original. Os metaprogramas usados pelo compilador de Cyan devem ser implementados usando o protocolo de metaobjetos (biblioteca) fornecido junto ao código-fonte do compilador da linguagem Cyan.

Nessa pesquisa foi desenvolvido um metaprograma em Cyan que durante a compilação gera uma versão replicável de aplicações que não implementam replicação. Com base em marcações (anotações) encontradas no código-fonte dessas aplicações não replicadas o compilador chama os metaprogramas desenvolvidos para gerar uma versão replicável da mesma aplicação. Essa nova versão usa o *framework* Treplica para implementar a replicação nessas aplicações.



Outro metaprograma que foi desenvolvido nessa pesquisa valida se o código-fonte não replicado apresenta casos de não determinismo. Essa validação não cobriu todos as possíveis situações de não determinismo e necessita de uma configuração prévia para poder identificar essas ocorrências. Essa configuração mapeia quais métodos podem ser considerados não deterministas, com base nessa lista o metaprograma visita os métodos implementados no código-fonte da aplicação procurando por chamadas a essas métodos não deterministas listados.

Detalhes de implementação como códigos-fonte exemplo podem ser encontrados no texto da dissertação [Ugliara 2018]. Nela também são mostrados outros *frameworks* usados para implementar replicação e outras linguagens de programação que tem suporte a outras formas de metaprogramação.

3. Contribuição e Trabalhos Futuros

Nesse trabalho foi proposto o uso da metaprogramação para permitir a implementação de requisitos não funcionais de uma aplicação de modo mais transparente. Essa proposta foi validada no contexto de aplicações replicáveis. Trabalhos futuros podem usar a mesma proposta em contextos diferentes como segurança, portabilidade, desempenho, usabilidade, padrões de código-fonte, e outros.

A validação aplicada ao caso de não determinismo mostra uma outra forma possível de utilizar a metaprogramação. Ela pode ser usada no intuito de verificar o código-fonte das aplicações na busca de erros conceituais de programação, que não seriam indicados pelo processo padrão de compilação. Essa abordagem também pode ser aplicada a problemas de outras áreas da computação, para identificar possíveis erros no uso de bibliotecas e *framework*.

A metaprogramação em compilação é um conceito que pode ser usado para padronizar atividades de programação realizadas de modo manual por equipes de desenvolvimento. Muitas implementações e verificações que hoje são realizadas de forma manual podem ser automatizadas em linguagens que permitem esse tipo de metaprogramação.

Durante a realização dessa pesquisa foi submetido um artigo, *Transparent Replication Using Metaprogramming in Cyan* [Ugliara et al. 2018], ao Simpósio Brasileiro de Linguagens de Programação (SBLP), que aconteceu em Fortaleza (Ceará, Brasil). Nesse artigo é mostrada a automatização do desenvolvimento do código-fonte. Ele não trata da verificação desse código-fonte. A automatização foi aplicada em programas distribuídos que foram desenvolvidos em Cyan usando Treplica. Essa publicação do ponto de vista dos detalhes técnicos está situada entre esse artigo a dissertação *Replicação Orientada a Metaprogramação* [Ugliara 2018].

O código-fonte desenvolvido nessa pesquisa está disponível na internet em um repositório de projetos (https://bitbucket.org/fellipe-ugliara/mestrado/src/master/). Caso seja necessário reproduzir essa pesquisa, o ponto de partida é esse repositório. Para os entusiastas do LaTeX, também estão disponíveis: o código-fonte dessa dissertação, e o código-fonte da apresentação usada na defesa desse trabalho. Os arquivos README.md do repositório contêm detalhes de como executar esse projeto.

Referências

Damaševičius, R. and Štuikys, V. (2015). Taxonomy of the fundamental concepts of metaprogramming.

Guimarães, J. d. O. (2018a). The cyan language. Technical report.

Guimarães, J. d. O. (2018b). The cyan language metaobject protocol. Technical report.

Ugliara, F. a. (2018). Replicação orientada a metaprogramação.

Ugliara, F. a., Vieira, G. M. D., and Guimarães, J. d. O. (2018). Transparent replication using metaprogramming in cyan.

Vieira, G. M. D. and Buzato, L. E. (2008). Treplica: Ubiquitous replication.

Vieira, G. M. D. and Buzato, L. E. (2010). Implementation of an object-oriented specification for active replication using consensus. Technical report.