Especificação Formal da Linguagem Guaraná Utilizando a Notação Z

Mauri J. Klein[1], Sandro Sawicki, Fabrícia Roos-Frantz, Rafael Z. Frantz
UNIJUÍ, Departamento de Ciências Exatas e Engenharias
Rua do Comércio, 3000. 98700-000, Ijuí – RS – Brasil
{mauri.klein, sawicki, frfrantz, rzfrantz}@unijui.edu.br

Palavras-chave: Integração de Aplicações Empresariais; Tolerância a Falhas; Notação Z.

Atualmente grande parte das corporações, contam com aplicações em seu ecossistema de software como suporte para os seus processos de negócio [9]. Estes são compostos, em sua maioria, por aplicações legadas, pacotes adquiridos de terceiros ou desenvolvidos especificamente para resolver um problema particular, o que dificulta a sua reutilização. Os ecossistemas de software são sistemas heterogêneos compostos por aplicativos que, normalmente, não foram projetados levando em conta a sua integração. A integração, porém, é necessária, principalmente porque permite a reutilização de duas ou mais aplicações, com o objetivo de otimizar processos de negócios já existentes ou fornecer suporte a novos processos.

A *Integração de Aplicações Empresariais* (EAI) fornece metodologias e ferramentas para desenvolver e implementar soluções de integração. O objetivo de uma solução de integração é manter dados sincronizados entre diferentes aplicações ou desenvolver novas funcionalidades sobre as já existentes [7].

Nos últimos anos, a demanda pela integração tem motivado o surgimento de ferramentas para o projeto e implementação de soluções de integração, como Camel [8], Spring Integration [3], Mule [2] e Guaraná [5]. Estas ferramentas facilitam a construção de soluções. No entanto, semelhante às aplicações convencionais, uma solução de integração pode apresentar falhas em qualquer parte de sua execução.

Um problema no software pode ser permanente (defeito) ou transiente (recurso como rede ou serviço temporariamente indisponível) e pode ou não ocorrer durante uma execução. Caso ocorra, gera-se um erro levando o sistema para um estado que, se não for tratado, pode resultar em um erro que então será perceptível aos usuários finais [1] [11].

Neste caso, uma característica desejável em ferramentas de integração é a Tolerância a Falhas (TF). Assim a solução de integração continuará funcionando corretamente apesar da ocorrência de falhas.

São quatro os estágios que envolvem a Tolerância a Falhas. O *Event Reporting*: reporta eventos sobre o funcionamento de uma solução de integração; *Error Monitoring*: monitora e detecta possíveis erros emitindo notificações sobre sua ocorrência; *Error Diagnosing*: analisa as notificações para conhecer as causas do erro e as partes da solução que foram afetadas;

[1] Bolsista CAPES/PROSUP

Error Recovery: tenta-se recuperar do erro [4]. Estes estágios são fundamentais para que a solução de integração apresente uma confiabilidade em sua execução. As tecnologias para construir soluções de integração como Camel, Spring Integration e Mule, fornecem um mecanismo de detecção de erro baseados principalmente na utilização de *try-catch* [6]. Dentre as propostas que estudamos, Guaraná é a única que proporciona um mecanismo de detecção de erro com base em um sistema de monitoramento que pode ser configurado usando uma linguagem baseada em regras [4].

Neste contexto, para validar a cobertura das regras escritas pelos Engenheiros de Software ou gerá-las automaticamente, este trabalho propõe especificar formalmente, através da Notação Z, a sintaxe abstrata, primeiro passo para a formalização do Guaraná DSL.

Na literatura existem inúmeros trabalhos que utilizam a especificação formal para modelar linguagens, dos quais destacamos a formalização de diagramas UML utilizando a Notação Z [10].

Contudo, a Notação Z é utilizada para a especificação formal de sistemas e baseia-se em princípios da matemática discreta e da lógica de primeira ordem. Emprega estruturas matemáticas como: conjuntos, relações e funções como forma de expressar o estado, comportamento e propriedades de um sistema. Possui sintaxe e semântica precisas que resultam em notações completas, coerentes, concisas e legíveis capazes de validar simbolicamente as propriedades do sistema sem a necessidade de compilá-los e testá-los.

Outra característica importante da especificação em Z, é que a mesma e formada por esquemas que são decomposições da especificação em partes menores e que podem ser combinados e usados em outros esquemas. Além disso, a notação Z utiliza-se da definição de tipos para cada objeto definido, sendo possível a checagem de tipos em Z.

Portanto, através da especificação formal utilizando a Notação Z, pode-se representar com clareza e precisão todos os tipos, esquemas e restrições do Guaraná DSL contribuindo para gerar automaticamente as regras da solução EAI e validá-las, permitindo verificar que sua especificação atende aos requisitos da solução de integração.

Referências

[1] Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. on Depend. and Secure Comp., Jan-Mar 2004

[2] Dossot, D. and D'Emic, J. (2009). Mule in Action. Manning.

[3] Fisher, M., Partner, J., Bogoevici, M., and Fuld, I. (2010). Spring Integration in Action. Manning.

[4] Frantz, R. Z., Corchuelo, R., and Molina-Jiménez, C. (2012). A proposal to detect errors in Enterprise Application Integration solutions. Journal of Systems and Software, 85(3):480–497.

[5] Frantz, R. Z., Quintero, A.M. R., and Corchuelo, R. (2011). A Domain-Specific Language to Design Enterprise Application Integration Solutions. Int. J. Cooperative Inf. Syst., 20(2):143–176.

[6] Goodenough, J. B. (1975). Exception handling: Issues and proposed notation. Communications of the ACM, 18(12):683–696.

[7] Hohpe, G. and Woolf, B. (2003). Enterprise Integration Patterns - Designing, Building, and Deploying Messaging Solutions. Addison-Wesley.

[8] Ibsen, C. and Anstey, J. (2010). Camel in Action. Manning.

[9] Messerschmitt, D. and Szyperski, C. A. (2003). Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press.

[10] Mostafa, A. M., Ismail, M. A., Bolok, H. E., and Saad, E. M. (2007). Toward a Formalization of UML2.0 Metamodel using Z Specifications. In Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on, volume 1, pages 694–701.

[11] R. Campbell and B. Randell. Error recovery in asynchronous systems. IEEE Trans. Soft. Eng., 1986