Como o formato de arquivos XML evolui? Um estudo sobre sua relação com código-fonte

David S. França[1], **Eduardo M. Guerra**[1], **Maurício F. Aniche**[2]

[1]Laboratório de Matemática e Computação Aplicada
Instituto Nacional de Pesquisas Espaciais (INPE) – São José dos Campos, SP – Brasil

[2]Departamento de Ciência da Computação
Universidade de São Paulo (USP), São Paulo - Brasil

{davidsfranca,guerraem,mauricioaniche}@gmail.com

Abstract. *There are different ways to integrate applications. A popular approach is the exchange of messages in XML format where both applications share the XML schema. This model, known as "contract", may change over time as structural modifications such as additions of new information. Changes like that propagate to the client applications, as they have to fit to the new structure and contract. The aim is to better understand how are the dynamics of the contract changes in a software project by means of analyzing messages contracts changes and their impact on the source code of open source and industry applications.*

Resumo. *Existem diferentes formas de integrar aplicações. Uma delas é a troca de mensagens em formato XML, onde ambas aplicações compartilham o modelo estrutural do formato daquele XML. Este modelo, chamado de "contrato", pode sofrer alterações ao longo do tempo, como modificações estruturais ou agregação de novas informações, fazendo com que as aplicações que o utilizam tenham que se adequar a nova estrutura sempre que o contrato é alterado. O objetivo é entender melhor como é a dinâmica de alterações de contratos em arquivos XML por meio de uma análise nas alterações de contratos de mensagens e seu impacto sobre o código-fonte de aplicações de código aberto e da indústria.*

1. Introdução

Com o crescente aumento da quantidade de informações disponíveis e a necessidade de troca desses dados entre aplicações, diversas soluções para integração de sistemas são utilizadas, adotando as mais diversas tecnologias encontradas no mercado. Segundo [Kalin 2013], é raro que um sistema de software funcione inteiramente de modo isolado. O típico sistema de software deve interagir com os outros sistemas, que podem estar em diferentes máquinas, serem escritos em diferentes linguagens, ou mesmo utilizarem diferentes plataformas.

Com esta finalidade de integração, umas das soluções é a utilização de arquivos *Extensible Markup Language*(XML) para troca de mensagens. A utilização desse de arquivo traz a vantagem de ser moldável às necessidades das aplicações, onde "elementos são arranjados em uma estrutura hierárquica, similarmente a uma árvore, que reflete tanto a estrutura lógica dos dados quanto a estrutura de armazenamento"[Bates 2003]. Muitas

vezes a integração entre as aplicações necessita de garantia na validade das informações, para isso, arquivos XML são validados segundo uma estrutura definida em um modelo estrutural de formato de dados que é compartilhado pelas aplicações chamado também de contrato. [Meyer 1997] diz que contrato "é relação entre uma classe e seus clientes como um acordo formal, expressando direitos e obrigações de cada parte". O padrão mais utilizado para a criação contratos de validação de arquivos XML é o *XML Schema Definition* (XSD). Nele são descritos, por meio de sua linguagem própria, qual a exata estrutura que um XML deve ter para ser considerado válido como, por exemplo, nomes das entidades, sua multiplicidade e opcionalidade. [Valentine 2002] dizem que, em outras palavras, o *schema* define as restrições para um vocabulário XML.

No entanto, as aplicações que compartilham um modelo na troca de mensagens podem sofrer alterações ao longo do desenvolvimento do projeto, fazendo com que as outras aplicações, que compartilham o mesmo modelo, não consigam mais estruturar suas mensagens no formato acordado. Para tanto, a cada mudança no modelo de mensagens todas as aplicações que o utilizam devem alterar suas implementações para criar mensagens no formato. [França and Guerra 2013] dizem que a evolução de aplicações quando um contrato é alterado dá-se de forma lenta e difícil, alterando-se apenas parte das aplicações por vez ou em velocidades diferentes. Assim, este trabalho tem por objetivo fazer um estudo acerca das mudanças nos contratos compartilhados por aplicações que desejam se integrar e qual é a relação de uma mudança desse contrato no código-fonte das aplicações. Para isso, quatro questões de pesquisa foram investigadas a fim de tentar entender a utilização de modelos de mensagens e sua relação com o código-fonte das aplicações:

(Q1) Qual é a frequência de modificações em arquivos XSD?

(Q2) Quais os tipos mais comuns de modificações em arquivos XSD?

(Q3) Qual a relação entre alterações em arquivos XSD e alterações no código-fonte?

(Q4) As alterações do XSD costumam ficar concentradas em um desenvolvedor?

Este artigo apresenta na seção 2 a metodologia de pesquisa, na seção 3 o estudo sobre projetos industriais, na seção 4 o estudo sobre os projeto de código aberto. Na seção 5 faz-se uma análise entre os dois estudos. Na seção 6 apresentam-se os trabalhos relacionados. Na seção 7 apresentam-se as conclusões e nas seções 7 e 8 apresentam-se as limitações encontradas na pesquisa e as conclusões do estudo..

2. Metodologia de pesquisa

O objetivo do estudo é extrair métricas a partir dos *commits* de projetos e analisar as alterações feitas em arquivos XSD, bem como sua relação com mudanças no código fonte. A análise dos dados tenta responder uma série de questões pertinentes às mudanças de arquivos XSD como quantidade de alterações, frequência de alterações, tipos de mudanças efetuadas, número de desenvolvedores que alteraram arquivos XSD e a relação dessas alterações com arquivos de código-fonte.

Para isso foram feitos dois estudos em dois grupos de projetos: industriais e de código fonte aberto. Inicialmente, foram adquiridos os dados referentes aos *commits* de

cada projeto através dos dados armazenados nos controladores de versões dos projetos. Utilizaram-se duas formas de obtenção de dados: manualmente, onde cada *commit* de alteração de arquivos de contratos foi avaliado; e outra de forma automática, utilizando uma ferramenta de mineração de dados. Para cada *commit* de alteração de arquivos de contrato, foi obtida a data de alteração, quem o alterou, o número de arquivos de código-fonte no mesmo *commit*, além das médias de alterações de arquivos XSD por projeto e a média dos arquivos de código-fonte alterados no mesmo *commit* de arquivos XSD.

Para tipos de mudança, em cada alteração no contrato verificou-se, comparando o arquivo antes e depois, o que mudou nele, podendo assim, distinguir qual foi a alteração sofrida e dividi-las quanto ao tipo de alteração. Para o caso da frequência de modificações, foi extraído de cada *commit* sua data, gerando uma série temporal de mudanças para cada arquivo. Quanto a sua relação com o código fonte, quando um *commit* possui um arquivo XSD, contou-se o número de arquivos de código-fonte também neste mesmo *commit*. Vale ressaltar que no caso da análise da relação com o código-fonte nos projetos industriais, esta contagem foi feita de forma manual, verificando no momento da recuperação de cada *commit* se os arquivos de codígo-fonte tem relação coma mudança no XSD. Esta análise foi feita de modo empírico, por desenvolvedores das aplicações. Por fim, comparou-se os dois conjuntos de métricas na esperança de se obter evidências a respeito do comportamento de tais arquivos independentemente do projeto avaliado. Essa comparação abrangeu a média de arquivos de código-fonte ligados a mudança de contrato, a frequência de modificações e a distribuição dos integrantes da equipe do projeto.

3. Estudo sobre projetos industriais

Fez-se um levantamento em três projetos que utilizam um mesmo modelo na troca de mensagens. Todos eles são gerenciados pelo controlador de versões *Subversion*. Um dos projetos é o fornecedor dos serviços, que provê dados às outras aplicações. Quanto aos projetos analisados, tratam-se de 3 aplicações da área de defesa que formam um simulador de eventos discretos com aproximadamente 5 anos de desenvolvimento. Na tabela 1 pode-se verificar alguns dados referentes a elas.

Tabela 1. Dados dos projetos industriais analisados

Nome	linhas de código	classes	XSDs	entidades nos XSDs
Fornecedor 1	27199	235	1	114
Consumidor 1	174085	1243	1	114
Consumidor 2	116648	1240	1	114

3.1. Realização do estudo

O primeiro passo é extrair todos os *commits* referentes às mudanças nos arquivos XSD. Para se obter o histórico de *commits* de cada arquivo do contrato (um contrato pode estar dividido em vários arquivos) utilizou-se a ferramenta TortoiseSVN. A recuperação dos dados foi feita de forma manual já que são apenas 3 projetos e 1 contrato compartilhado entre eles. A comparação entre versões do contrato também foi feita de forma manual, isto é, para cada mudança no contrato, obtem-se a diferença antes e depois da alteração e

comparando-o para saber de que tipo de mudança se trata. Nesse passo 95 *commits* foram selecionados para análise.

Desses dados foram extraídas as frequências das modificações, divididas em trimestres, já que os projetos possuem um longo período de desenvolvimento, além da distribuição das mudanças dos contrato entre os desenvolvedores dos projetos. Além disso, uma alteração no contrato da mensagem influencia tanto o fornecedor, que deve alterar seu código-fonte para gerar a mensagem no novo formato, como os consumidores, que devem alterar seus códigos-fonte para consumirem as mensagens no novo formato. Assim foram também extraídos dados em relação ao número arquivos de código-fonte alterados quando um contrato sofre modificações.

4. Estudo sobre projetos de código aberto

A segunda etapa consiste em obter e analisar dados de um repositório de projetos de código aberto com um maior volume de dados. É feito um levantamento em um repositório da USP (Universidade de São Paulo) gerenciado pelo controlador de versões Git, possuindo 67 projetos, dentre eles 22 utilizam diferentes contratos. Para este estudo é utilizada a ferramenta MetricMiner [Sokol et al. 2013], uma aplicação *web* que facilita o trabalho de pesquisadores envolvidos em mineração de dados em repositórios de software. A ferramenta ajuda pesquisadores em todas as etapas durante um estudo de mineração de repositório de código, como a clonagem do repositório, extração de dados, criando conjuntos de dados específicos e executar testes estatísticos. Utilizando as funcionalidades desta ferramenta, foi desenvolvida uma consulta e acoplada a ferramenta de mineração, que retorna apenas os dados avaliados como relevantes na análise.

4.1. Realização do estudo

Foram extraídas métricas referentes ao número de arquivos XSD presentes nos projetos, a frequência em que foram alterados e a distribuição dessas alterações ao longo do tempo. Dos 67 projetos avaliados, 22 contêm arquivos XSD e destes, 11 os alteraram. Foram extraídos tambem dados referentes a média de alterações por mudança em um XSD, além do número de arquivos de código-fonte alterados a cada mudança em um XSD.

5. Análise dos resultados

(Q1)Qual é a frequência de modificações em arquivos XSD? Os estudos indicam que quando um projeto possui arquivos XSD e este é alterado, não ocorre apenas uma vez, mas várias vezes e distribuídos ao longo do período de desenvolvimento do projeto. Isto indica a necessidade de atualização de todas as aplicações que precisam seguir o contrato alterado. No primeiro estudo, dos 19 trimestres analisados, apenas 5 não tiveram alterações no contrato, isto mostra que as outras duas aplicações que compartilham este arquivo, também foram ou deveriam ser atualizadas constantemente para continuar seguindo o contrato, dependendo da necessidade de utilização da nova imformação disponivel pelo contrato e poderem trocar informações entre si, como mostra a Figura 1. No segundo estudo, viu-se que em alguns projetos as modificações chegam a acontecer durante todo o desenvolvimento, como mostra a Figura 2. Alterá-los e replicar essas mudanças em outras aplicações que o utilizam pode ser muito custoso ou até mesmo inviável.

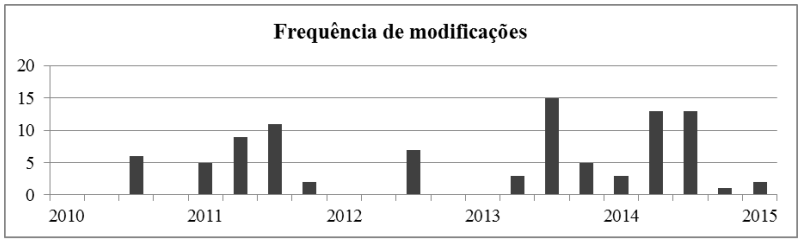

Figura 1. Frequência das modificações nos contratos XSD

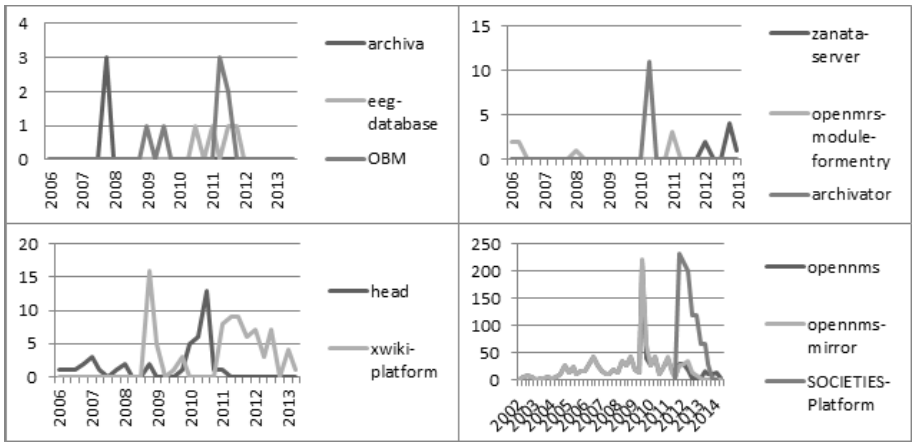

Figura 2. Alteração de arquivos XSD por projeto ao longo do desenvolvimento

(Q2)Quais os tipos mais comuns de modificações em arquivos XSD? Observa-se no primeiro estudo 9 tipos de modificações nos arquivos XSD, onde a adição de um novo elemento e alteração de um elemento são mais comuns como é mostrado na Tabela 2, indicando respectivamente, a inserção de novos dados a uma mensagem e alterações de estrutura ou semântica. Logo, estes tipos de modificações podem quebrar o contrato de mensagens entre as aplicações.

(Q3)Qual a relação entre alterações em arquivos XSD e alterações no código-fonte? Verificou-se que contratos têm certa relação com seu código-fonte. Para cada modificação no contrato é necessário um esforço para adequar a aplicação para ler ou gerar mensagens no novo formato. Pode ser muito custoso quando várias aplicações estão envolvidas ou a equipe é pequena. Vê-se no segundo estudo, que a média de modificações de arquivos de código-fonte quando um contrato é alterado pode ser muito alta, indicando esse tipo de relação, mostrado na Tabela 3. Como segunda evidência deste relação, no primeiro estudo, mostrado na Figura 3, para cada *commit* de alteração no contrato foi verificado classe a classe do código-fonte no mesmo *commit* se esta estava ligada ou não a mudança do contrato. Essa verificação é feita de forma empírica, baseado na experiência dos desenvolvedores do projeto, para se dizer se a alteração da classe está ou não ligada a mudança no contrato. Como uma mudança em um contrato pode não sofrer *commit* juntamente com as alterações no código-fonte, foram considerados os 5 *commits* adjacentes à alteração do contrato, para se tentar garantir que as alterações de código-fonte estavam sendo corretamente contadas. Constatou-se através dessa análise manual que, em média, 43,77% dos arquivos de código-fonte no mesmo *commit* de alteração do contrato estão relacionados a mudanças nos contratos.

Tabela 2. Tipos de Modificações em arquivos XSD

Tipo de Modificação	Quantidade
Adição de um novo elemento	51
Modificação de um elemento	18
Alteração de multiplicidade	15
Alteração de enum	12
Modificação de um atributo	10
Remoção de um elemento	5
Adição de enum	3
Adição de um novo atributo	2
Remoção de um atributo	2

Tabela 3. Estatísticas de modificações em projetos código aberto

Projeto	NAX	MAX	MMAX	MAJ	MMAJ	NC
archiva	1	2	2,00	15	7,50	1
archivator	1	7	7,00	9	1,29	4
eeg-database	1	2	2,00	61	30,50	2
head	5	29	5,80	1943	67,00	12
OBM	1	1	1,00	25	25,00	1
openmrs-module-formentry	2	3	1,50	18	6,00	2
opennms	212	2065	9,74	25743	12,47	37
opennms-mirror	154	2005	13,02	42121	21,01	36
SOCIETIES-Platform	104	1751	16,84	47836	27,32	54
xwiki-platform	5	76	15,20	7700	101,32	14
zanata-server	1	5	5,00	6337	1267,40	4

Legendas: NAX – Nº de Arquivos XSD, MAX – Modificações em Arquivos XSD, MMAX – Média de Modificações em Arquivos XSD, MAJ – Modificações em Arquivos JAVA, MMAJ - Média de Modificações em Arquivos JAVA, NC – Nº de Commiters.

(Q4)As alterações do XSD costumam ficar concentradas em um desenvolvedor? Como mostrado na Figura 4, notou-se uma diferença bastante relevante. Projetos industriais tendem a possuir a mesma equipe de desenvolvimento por um período maior que os de código aberto. Isso foi avaliado na obtenção dos dados, onde os projetos industriais analisados tiveram 11 desenvolvedores alterando o contrato durante os 5 anos de desenvolvimento dos projetos. Já nos projetos de código aberto, em média, tiveram 15,18 desenvolvedores alterando os contratos dos projetos em um tempo menor de desenvolvimento.

6. Trabalhos relacionados

Diferentemente do presente trabalho que trata de formatos de arquivos XML, [Su et al. 2001] relacionam as mudanças dos *XML schemas* de banco de dados XML com *queries* criadas e classifica as alterações tanto no *schema* quanto nas *queries*. No trabalho é dito que um *schema* pode ter nova versão até a cada 15 dias em um projeto e as

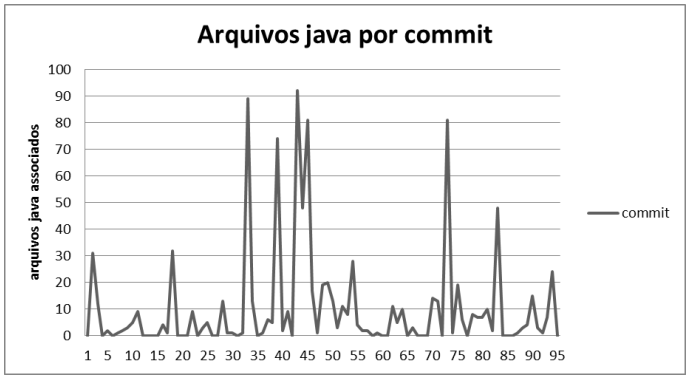

Figura 3. Alteração de arquivos .java por *commit* de alteração no contrato

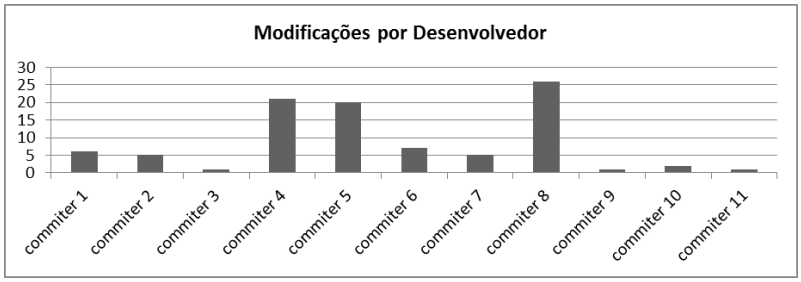

Figura 4. Número de alterações por desenvolvedor

mudanças no XML e nas *queries* são divididas em dois grupos: mudanças basicas e complexas. Por fim, concluiu propondo um conjunto inicial de diretrizes para a preservação de consultas em diferentes versões do esquema.

[Mesiti et al. 2006] fazem uma investigação das mudanças ocorridas em *XML schemas* classificando-as em tipos. Foi também apresentado um conjunto de mudanças primitivas que é a base para qualquer outro tipo de mundança. Esses tipos foram utilizados para propor uma solução de revalidação de XMLs contra *XML schemas* por partes, utilizando mudanças atômicas, livrando assim de ter que se validar tod O presente trabalho também faz uma investigação das mudanças ocorridas nos XML *Schemas*, porém também se preocupa investigar o impacto dessas mudanças sobre o código-fonte.

[Nečaskỳ et al. 2012] versam sobre a evolução de *XML schemas* quando há alteração de requisito. As modificações são mapeadas de acordo com seus espalhamentos horizontais e verticias sobre o sistema. Assim como [Mesiti et al. 2006], ele apresenta um conjunto de mudanças primitivas e o trabalho propõe um *framework* para esse evolução, abrangendo mundanças do tipo *platform-independent model* e *platform-specific model*. O trabalho se preocupa em rastrear as mudanças ocorridas nos modelos quando um requisito é alterado, diferentemente do presente estudo que continua esta linha, partido da alteração do XML *schema* e rastreado seu impacto no código-fonte.

7. Limitações do estudo

XSD de configuração: Uma das limitações do estudo foi considerar as definições de arquivos XML a partir de arquivos XSD de de forma equivalente em toda aplicação, independente do propósito do documento, como, por exemplo, para armazenamento em

disco ou para contratos de serviços *web*. **Poucos projetos**: outro ponto que se deve considerar é a pequena quantidade de projetos de cunho industrial, pois a aquisição de código fonte de projetos deste cunho muitas vezes não é trivial, já que envolvem produtos de empresas que necessitam manter certa privacidade. **Tipo de modificações em código aberto**: outra limitação foi a não investigação dos tipos de alterações dos contratos nos projetos de código aberto, pois demandaria um tempo muito grande neste refinamento da pesquisa. Uma possivel solução é avaliar as mudança pelos comentários dos *commits*.

8. Conclusões

Concluiu-se que são frequentes as modificações dos contratos em ambos os tipos de projetos, e as modificações mais encontradas alteram significantemente a estrutura da mensagem. Essas modificações não são pontuais, sendo diluídas ao longo do desenvolvimento dos projetos. O estudo apresentou indícios que essas modificações geram um forte impacto no código-fonte, fazendo com que uma mudança em um contrato implique na alteração de implementação dos projetos. Concluiu-se ainda que as alterações não estão concentradas em um único desenvolvedor, mas distribuídas por vários integrantes do time.

Quanto a relação de mudanças de XSD com alterações do código-fonte, ficou evidente, no primeiro estudo, que existe a relação de alteração do código-fonte quando um contrato é alterado. Isto complementado pelo segundo estudo que indicou que mudanças em contratos são frequentes ao longo do desenvolvimento.

Referências

Bates, C. (2003). *XML in theory and practice*. J. Wiley, Chichester, England Hoboken, NJ.

França, D. S. and Guerra, E. M. (2013). Modelo arquitetural para evolução no contrato de serviços no contexto de aplicações de comando e controle. *XV Simpósio Internacional de Guerra Eletrônica, SIGE*, 15:23.

Kalin, M. (2013). *Java Web services : up and running*. O'Reilly, Beijing.

Mesiti, M., Celle, R., Sorrenti, M. A., and Guerrini, G. (2006). X-evolution: A system for xml schema evolution and document adaptation. In *Advances in Database Technology-EDBT 2006*, pages 1143–1146. Springer.

Meyer, B. (1997). *Object-oriented software construction*. Prentice Hall PTR, Upper Saddle River, N.J.

Nečaskỳ, M., Klímek, J., Malỳ, J., and Mlỳnková, I. (2012). Evolution and change management of xml-based systems. *Journal of Systems and Software*, 85(3):683–707.

Sokol, F. Z., Finavaro Aniche, M., Gerosa, M., et al. (2013). Metricminer: Supporting researchers in mining software repositories. In *Source Code Analysis and Manipulation (SCAM), 2013 IEEE 13th International Working Conference on*, pages 142–146. IEEE.

Su, H., Kramer, D., Chen, L., Claypool, K., Rundensteiner, E., et al. (2001). Xem: Managing the evolution of xml documents. In *Research Issues in Data Engineering, 2001. Proceedings. Eleventh International Workshop on*, pages 103–110. IEEE.

Valentine, C. (2002). *XML schemas*. SYBEX, Alameda, Calif.