ProFap-R: Um Processo de Reengenharia de Código Orientada pela Reengenharia de Dados

Eduardo Moreira Fernandes, Ygo Aquino Brito, Inara Santana Ortiz, Nathalia Leite B. de M. Borine, Maria Istela Cagnin[1]

Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)
Caixa Postal 549 – 79.070-900 – Campo Grande – MS

{eduardomorfernandes, ygo1992, inara.ortiz, nathaliaborine}@gmail.com,
istela@facom.ufms.br

Abstract. *Legacy systems are computational systems that offer relevant services for organizations, but are considered obsolete because of their older technologies. Furthermore, they use to have high complexity and cost of maintenance. Aiming to evolve legacy systems, different techniques are used as the software reengineering. This work presents a process of code reengineering oriented by data reengineering for midrange systems. This process was successfully applied in the reengineering of a specific Web system of the domain of social management.*

Resumo. *Sistemas legados são sistemas computacionais que oferecem serviços de alta relevância para as organizações, porém considerados obsoletos por suas tecnologias ultrapassadas. Além disso, possuem geralmente alto grau de complexidade e elevado custo de manutenção. Com o intuito de evoluir sistemas legados são empregadas técnicas como a reengenharia de software. O presente trabalho apresenta um processo de reengenharia de código orientada pela reengenharia de dados para sistemas legados de médio porte. Esse processo foi aplicado com sucesso na reengenharia de um sistema Web específico do domínio de gestão social.*

1. Introdução

Sistemas legados são sistemas computacionais que oferecem serviços de alta relevância para organizações, mas são considerados obsoletos por conta de suas tecnologias ultrapassadas. Também são caracterizados por seu alto grau de complexidade e elevado custo de manutenção, devido a fatores como a falta de documentação disponível e a dificuldade de atualização das tecnologias empregadas no desenvolvimento do sistema original [Pressman 2011; Sommerville 2011].

Com intuito de modernizar sistemas legados e garantir manutenibilidade, são empregadas técnicas como a reengenharia de software [Chikofsky e Cross 1990], composta basicamente pelas etapas de engenharia reversa e engenharia avante. Na primeira, o sistema legado é representado por artefatos de alto nível de abstração. Na segunda, é feita a engenharia avante do sistema tomando como base os artefatos da etapa anterior. Além disso, a reengenharia de software provê documentação atualizada, renovação das tecnologias empregadas e reestruturação do sistema.

[1] Apoio financeiro da Fundect (Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul) - T.O. nº 0072/12.

A principal motivação deste trabalho foi a reengenharia do Sistema de Informação em Gestão Social (SIGS), o qual é um sistema Web de médio porte com desenvolvimento iniciado no ano 2000 pelo Instituto de Estudos Especiais (IEE) da Pontifícia Universidade Católica de São Paulo (PUC-SP). Esse sistema é considerado obsoleto principalmente pelas tecnologias ultrapassadas com as quais foi implementado. A reengenharia do SIGS foi motivada durante a execução de um projeto de pesquisa voltado ao Sistema Único de Saúde (SUS) [Cagnin *et al.* 2012], e está sendo realizada no Laboratório de Engenharia de Software (LEDES) da Universidade Federal de Mato Grosso do Sul (UFMS).

Inicialmente se propôs a adição de novas funcionalidades ao SIGS, relacionadas a área da saúde, para obtenção de relatórios com informações sociais e de saúde para apoiar tomadas de decisão de gestores de ambas as áreas. No entanto, observou-se a inviabilidade disso devido à baixa qualidade do código do SIGS (código replicado e inutilizado, falta de padronização, *design* deteriorado por manutenções, etc.) e das tecnologias obsoletas empregadas. Então, optou-se pela reengenharia desse sistema.

Considerando-se a complexidade de portar o código legado do SIGS para uma linguagem moderna, foi conduzida uma pesquisa por processos de reengenharia para sistemas de médio porte cuja reengenharia do código fosse baseada na reengenharia de dados. Isto pois, segundo Pérez-Castillo *et al.* (2013), bancos de dados existentes não devem ser descartados durante a modernização de *software*, pois contêm valioso conhecimento do negócio não presente em outro lugar. No entanto, processos desse tipo não foram encontrados.

Assim, o processo colaborativo de manutenção de software ProFap [Cagnin *et al.* 2013], definido e utilizado no LEDES, foi adaptado para o contexto de reengenharia e é apresentado neste artigo. O processo resultante dessa adaptação, nomeado como ProFap-R, propicia a reengenharia de código baseada na reengenharia de dados com o intuito de viabilizar a reengenharia de sistemas de médio porte, como é o caso do SIGS.

A escrita deste artigo está organizada em mais sete seções. Na Seção 2 são apresentados alguns trabalhos relacionados a processos de reengenharia. Na Seção 3 é apresentado o sistema SIGS legado. Nas Seções 4 e 5 são apresentados, respectivamente, o ProFap e o ProFap-R. Na Seção 6 é descrita a condução da reengenharia do SIGS com o apoio do ProFap-R. Por fim, na Seção 7, é feita uma conclusão acerca do presente trabalho.

2. Trabalhos Relacionados

Processos de reengenharia de software têm sido definidos nos últimos anos para modernizar sistemas legados desenvolvidos em plataforma *desktop* ou Web para a plataforma web baseada ou não em serviços, utilizando tecnologias atuais.

Ruiz *et al.* (2012) apresentam um arcabouço de reengenharia que a partir do sistema legado (*as-is system*) obtém modelos em um nível mais alto de abstração (*as-is models*), caracterizando a engenharia reversa. Em seguida, melhorias podem ser incorporadas nesses modelos para adequá-los às necessidades atuais da organização (*to-be models*). Após isso, a engenharia avante do sistema é realizada com base nos modelos *to-be* e em um modelo de processo de desenvolvimento de software (por exemplo, cascata, espiral, incremental, etc.), obtendo-se o sistema pós-reengenharia (*to-be system*). Rodríguez-Echeverría *et al.* (2011) apresentam um arcabouço para a

modernização sistemática de aplicações Web para aplicações RIAs[2] (*Rich Internet Applications)*. Basicamente, o principal objetivo do arcabouço proposto é gerar uma aplicação cliente RIA a partir das camadas de navegação e de apresentação da aplicação Web legada e permitir a comunicação entre a sua camada de conexão orientada a serviço com a camada da lógica do negócio do lado do servidor. Por outro lado, Pérez-Castillo *et al.* (2013) propõem um processo de reengenharia que, em linhas gerais, recupera serviços Web a partir de bases de dados legadas. Os serviços Web identificados gerenciam acesso às bases de dados legadas sem descartá-las. Com isso, bases de dados legadas podem então ser usadas por sistemas de informação modernizados em ambientes orientados a serviços.

No entanto, pelas buscas realizadas não foram encontrados trabalhos direcionados à reengenharia de sistemas de médio porte nos quais a reengenharia de dados conduz a reengenharia de código, que é o foco deste trabalho.

3. SIGS Legado

O SIGS é um sistema de informação de âmbito social que oferece um controle de dados eficaz, visando o auxílio do monitoramento, da sistematização e da avaliação de programas sociais. Em plataforma Web, oferece às prefeituras e às instituições sociais um conjunto de serviços como o cadastro e o acompanhamento das famílias incluídas nos programas. O SIGS é somente acessado pelos profissionais das instituições que o utilizam, e não pela população em geral, havendo rigoroso controle de acesso por permissões de usuários.

Inicialmente, o SIGS foi orientado à gestão de programas sociais estaduais de complementação de renda pela Secretaria de Assistência Social da Prefeitura de São Paulo, de 2002 a 2004. Em seguida, passou a ser utilizado em vários programas no Estado de MS a partir de uma parceria realizada entre a PUC e a UFMS: Vale Universidade, Vale Universidade Indígena, Unidades Socioeducativas, dentre outros.

No contexto do uso do SIGS, agentes sociais são responsáveis pela coleta de dados obtidos junto à população, e os dados são cadastrados no SIGS por técnicos responsáveis. Depois do registro dos dados, relatórios gerenciais são fornecidos pelo sistema. O SIGS foi desenvolvido utilizando as tecnologias *ASP* e SGBD (Sistema de Gerenciamento de Banco de Dados) *SQL Server 2000*, que são proprietárias e atualmente obsoletas. Esse sistema pode ser considerado de médio porte pois possui 498.769 LOC e 383 tabelas.

4. Processo ProFap

O ProFap, proposto por Cagnin *et al.* (2013), consiste de um processo colaborativo de manutenção com integração de ferramentas de apoio a diversas atividades e foco no desenvolvimento colaborativo de *software*. Ele tem sido amplamente utilizado por diversos projetos do LEDES, em especial, o projeto do SIGFAP (Sistema de Informação de Gestão de Fundações de Amparo à Pesquisa), que atualmente está sendo utilizado por Fundações de Amparo à Pesquisa de doze estados do país. O ProFap é

[2] RIAs possuem plataforma baseada na Web 2.0 e oferecem principalmente interfaces de usuário com alto nível de usabilidade e interação, e processamento e armazenamento de dados no lado do cliente.

dividido em etapas, cada uma com procedimentos específicos e possui um fluxo conforme ilustrado na Figura 1.

Após a implantação da versão inicial do ProFap, diversas propostas de melhorias foram identificadas pelos membros do LEDES. Uma dessas melhorias consiste na adição da etapa "Projetar Solução", explicada mais à frente. Assim, o ProFap após as melhorias é composto de seis etapas, conforme apresentadas na Figura 1.

Para apoiar as etapas do ProFap, faz-se uso de ferramentas de desenvolvimento de *software* integradas, tais com o Redmine (para gerenciamento de projetos e *bugs*) e o Git (sistema de versionamento de código e artefatos).

Na etapa "Solicitar Manutenção", é feita a solicitação via Redmine de novas funcionalidades do sistema ou o reporte de *bugs*. Se a solicitação for relativa a manutenção corretiva, então será atribuída a um desenvolvedor. Se referente ao desenvolvimento de módulo ou funcionalidade inédita, então será avaliada por comitês na etapa "Aprovar Solicitação de Manutenção". Nessa etapa, uma solicitação pode ser aprovada ou cancelada.

Na etapa "Projetar Solução", após aprovação de uma solicitação, esta é atribuída a um desenvolvedor. Nela, desenvolvedores com tarefas atribuídas relacionadas entre si discutem a melhor forma de projetar a solução para atender tais tarefas, com auxílio dos líderes de equipe que possuem conhecimento detalhado do domínio do sistema. Em paralelo à etapa "Projetar Solução", ocorre a etapa "Implementar Solução e Efetuar Testes de Unidade". Nessa etapa, a solução projetada na etapa anterior é implementada com auxílio da ferramenta Git. Testes funcionais e unitários devem ser feitos pelo desenvolvedor, em seu Ambiente Local, durante a implementação da solicitação. Nessas etapas, pode ocorrer via Redmine comunicação entre cliente e desenvolvedor.

Na etapa "Efetuar Testes de Integração e de Aceitação" é feita atualização do ambiente de testes, via Git integrado ao Redmine, com o resultado do trabalho. Após isso, são feitos testes de integração no ambiente de testes. O desenvolvedor poderá solicitar apoio de outras equipes de desenvolvimento para os testes de aceitação. Esse tipo de teste é sempre feito pelo cliente. Por fim, na etapa "Finalizar Tarefa de Manutenção", ocorre o encerramento da solicitação de manutenção via Redmine e uma nova versão do sistema é atualizada no ambiente de produção utilizando o Git.

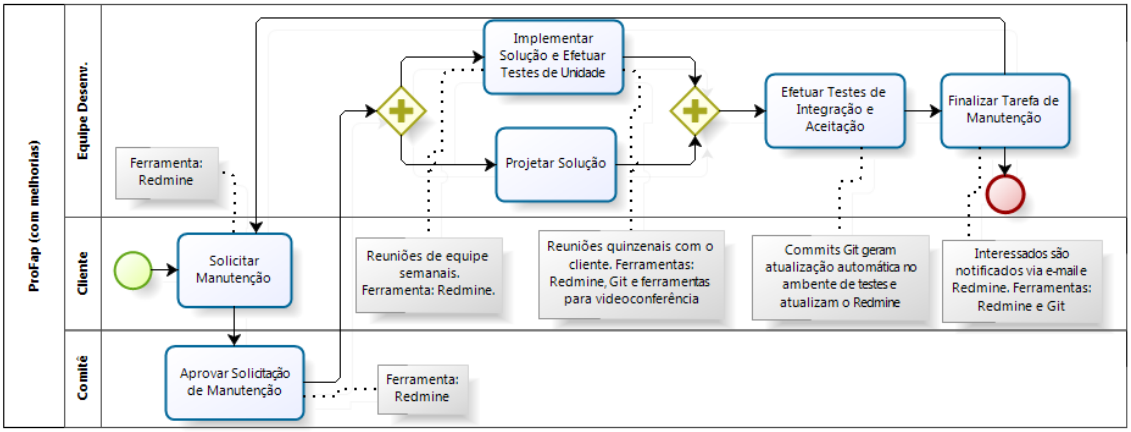

Figura 1. ProFap: Processo Colaborativo de Manutenção (adaptado de Cagnin *et al.* (2013))

5. Processo ProFap-R

O processo proposto neste artigo, ProFap-R, é uma adaptação do ProFap para apoiar a reengenharia de código orientada pela reengenharia de dados. Para sua elaboração, foram concebidas novas etapas em relação ao ProFap, e outras foram adaptadas de acordo com as necessidades da reengenharia, conforme a Figura 2. As mesmas ferramentas de apoio utilizadas no ProFap (Figura 1) podem ser utilizadas no ProFap-R.

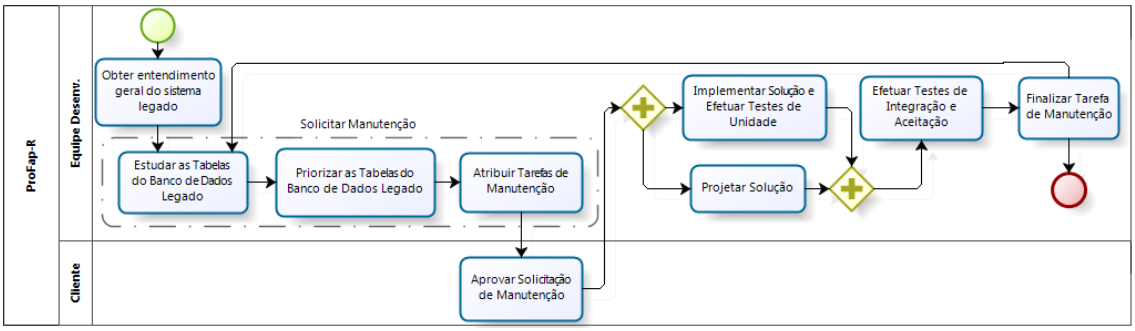

Figura 2. ProFap-R: Processo de Reengenharia de Código Orientada pela Reengenharia de Dados

Na etapa "Obter entendimento geral do sistema legado", é obtido um amplo entendimento do domínio do sistema e são decididas as tecnologias que serão utilizadas na reengenharia. A etapa "Solicitar Manutenção" do ProFap foi desdobrada em três etapas: i) "Estudar as tabelas do banco de dados legado": a equipe de desenvolvimento realiza um entendimento do sistema legado e documenta as suas tabelas; ii) "Priorizar as tabelas do banco de dados legado": as tabelas documentadas são priorizadas das menos às mais dependentes. Essa priorização determinará a ordem de submissão à reengenharia das tabelas e funcionalidades associadas a elas, a fim de facilitar a migração do sistema com menos esforço; iii) "Atribuir tarefas de manutenção": tarefas de manutenção são atribuídas aos desenvolvedores de acordo com a priorização realizada. O entendimento de cada funcionalidade e das regras de negócio associadas é baseado na execução do sistema legado e descrito na própria tarefa.

Na etapa "Aprovar Solicitação de Manutenção", é necessário que as funcionalidades submetidas à reengenharia sejam aprovadas pelo cliente, para que funcionalidades não representativas dos processos de negócio atuais não sejam encaminhadas à reengenharia. Caso o cliente não esteja presente e/ou disponível, essa etapa é desconsiderada. Na etapa "Projetar Solução", são avaliadas as tabelas do sistema legado relacionadas à solicitação de manutenção, a fim de definir as tabelas do sistema pós-reengenharia. Para isso, são: eliminadas redundâncias das tabelas do sistema legado seguindo regras de normalização de bancos de dados; padronizados os nomes dos campos; definidos os tipos de dados correspondentes ao novo SGBD adotado, se for o caso, entre outros. Caso algum *framework* de aplicação [Fayad e Johnson 2000] no domínio do sistema legado tenha sido selecionado para apoiar a reengenharia, o projeto das funcionalidades deve ser baseado no projeto do mesmo.

Na etapa "Implementar Solução e Efetuar Testes de Unidade", as tabelas associadas à manutenção são migradas para o novo SGBD. Em alguns casos o *script* de criação do banco de dados legado pode ser reutilizado totalmente ou parcialmente. Porém, nos casos em que o banco de dados não está normalizado e sem padronização nos nomes de tabelas, campos, chaves primárias e estrangeiras, sugere-se escrever um novo *script*. As funcionalidades são implementadas de acordo com o projeto definido na etapa anterior e com base na descrição das regras de negócio realizada na etapa

"Atribuir tarefas de manutenção". Caso algum *framework* de aplicação no domínio do sistema legado tenha sido selecionado, o mesmo é utilizado para apoiar a implementação. Na etapa "Implementar Solução e Efetuar Testes de Unidade" dados de tabelas do sistema legado podem ser reaproveitados para povoar tabelas do sistema pós-reengenharia. As etapas "Efetuar Teste de Integração e Aceitação" e "Finalizar Tarefa de Manutenção" do ProFap-R são similares ao processo ProFap.

6. Reengenharia do SIGS

O processo ProFap-R foi utilizado na reengenharia do SIGS, a qual foi realizada por bolsistas (assumindo o papel de equipe de desenvolvimento do processo) do projeto de pesquisa mencionado na Seção 1, com o auxílio de um dos clientes envolvido no desenvolvimento do SIGS (assumindo o papel de cliente do processo). No início, foi executada uma análise das funcionalidades básicas do SIGS legado e a quantidade de tabelas no banco de dados, visando definir as tecnologias a serem empregadas na reengenharia. Na Figura 3 são ilustradas as arquiteturas do SIGS legado e do SIGS pós-reengenharia.

A arquitetura do SIGS legado (Figura 3 (a)) não seguiu padrões de projeto e há vários trechos de código não utilizados. Além disso, o SIGS legado possui base de dados única, acessada por várias instituições que oferecem recursos e programas sociais. Dependendo da quantidade de acessos simultâneos ao sistema e à sua base de dados, pode ocorrer queda do servidor ou aumento do tempo de resposta às consultas, além da possibilidade de colocar em risco a segurança das informações, caso as regras de segurança sejam violadas.

Na reengenharia do SIGS, foi estabelecido o uso de tecnologias multiplataformas e livres, bem como mudanças na arquitetura, como mostrado na Figura 3 (b).

Considerando a facilidade que *frameworks* de aplicação oferecem à engenharia avante de sistemas legados [Cagnin *et al*. 2003], foi utilizado neste trabalho o Titan Framework, que utiliza linguagens PHP e XML, para apoiar a reengenharia do SIGS. Esse pertence ao domínio *e-gov* [Carromeu *et al*. 2010] e utiliza o padrão de projeto MVC (*Model-View-Controller*). Basicamente, o Titan recebe como entrada arquivos de configuração (XML e SQL) da instância. Com isso, ele possibilita configurar as regras de negócio da nova aplicação editando o arquivo XML e, se isso não for suficiente, é possível programar novos componentes de software que podem ficar disponíveis ou não no repositório do *framework*, dependendo da sua generalização. O uso do Titan ofereceu diversas vantagens como maior controle de permissões dos usuários, organização do projeto em camadas MVC (explicitadas na Figura 3 (b) por retângulos com diferentes cores de cinza) e boa usabilidade da interface com o usuário.

Com a arquitetura estabelecida para a reengenharia, cada instituição possui uma instância do SIGS, consequentemente, o banco de dados é exclusivo para cada uma delas, dirimindo os problemas do SIGS legado.

As atribuições de tarefas aos desenvolvedores e a documentação delas foram realizadas por meio da ferramenta Redmine na etapa "Solicitar Manutenção" e começaram após a primeira análise do sistema (etapa "Obter entendimento geral do sistema legado"). Essas atribuições foram definidas em reuniões semanais e de acordo com o grau de dependência das tabelas da base de dados legada, iniciando pelas tabelas

menos dependentes e finalizando pelas mais dependentes. Todas as atas das reuniões também foram adicionadas ao Redmine como tarefas do tipo "reunião".

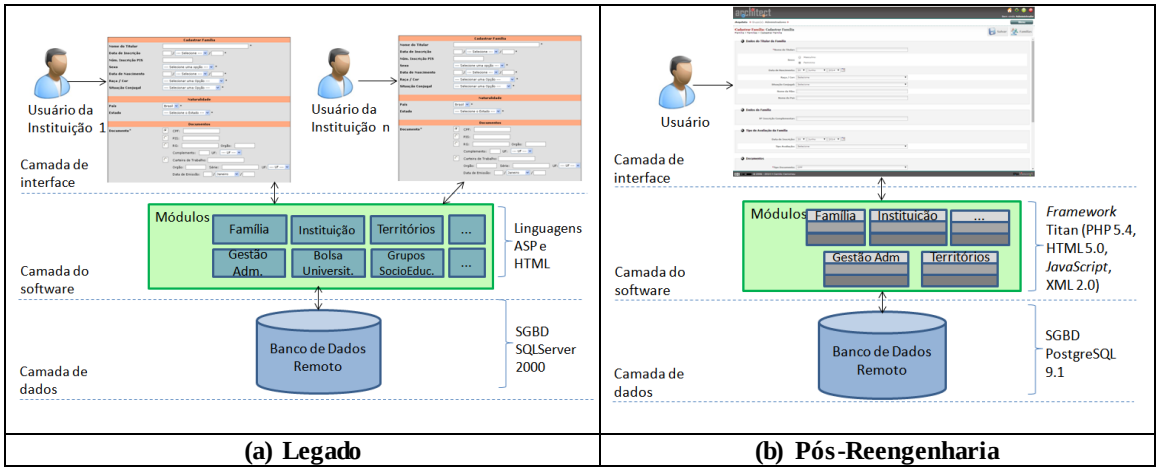

Figura 3. Arquiteturas do SIGS

Após as atribuições das tarefas, o cliente aprovou a realização das mesmas, como descrito na etapa "Aprovar Solicitação de Manutenção". Nessa etapa, algumas funcionalidades do SIGS legado foram rejeitadas para a reengenharia, pois não correspondiam mais às necessidades do cliente.

Durante a etapa "Projetar Solução", os desenvolvedores tomavam as decisões nas reuniões semanais onde as atribuições das tarefas eram realizadas. Soluções eram traçadas com base na descrição da tarefa e no projeto do Titan. Na fase "Implementar Solução e Testes de Unidade", a equipe de desenvolvimento criou os *scripts* das tabelas associadas a cada tarefa de manutenção e informou os parâmetros necessários no arquivo de configuração em XML do Titan. Logo após a implementação, foram realizados os testes unitários.

Na etapa "Efetuar Testes de Integração e de Aceitação" foram conduzidos testes de integração pela equipe de desenvolvimento e testes de aceitação em conjunto com o cliente, a fim de avaliar e garantir consistência e qualidade do produto. Após concluir cada tarefa, uma nova versão do produto era disponibilizada no ambiente de produção, de acordo com a etapa "Finalizar Tarefa de Manutenção". Foram contabilizadas 29.074 linhas de código (LOC) e 134 tabelas implementadas no SIGS pós-reengenharia, o que dá uma redução de 94% em LOC e 65% na quantidade de tabelas. Essa redução deve-se à reestruturação do banco de dados após uma análise refinada (anteriormente, o banco de dados atendia cada instituição onde o SIGS era implantado), mudança da linguagem de programação utilizada e adoção de um *framework* com reaproveitamento de componentes, além da remoção de funcionalidades não mais necessárias. Em todas as etapas do ProFap-R foi elaborada documentação relacionada diretamente à reengenharia do SIGS e também à condução do processo em si, como tomadas de decisão, *feedbacks* do cliente e atas de reuniões. Toda documentação produzida foi registrada e versionada no Git via Redmine.

7. Conclusão

Este artigo apresentou o processo ProFap-R, que é um processo de reengenharia de código baseado na reengenharia de dados, podendo ser apoiado por *frameworks* do domínio do sistema legado. Esse processo permitiu com sucesso a condução da

reengenharia de um sistema de médio porte, com o apoio do Titan Framework por alunos de graduação não especialistas em reengenharia. Isso foi possível devido à simplicidade e flexibilidade tanto do processo proposto quanto das ferramentas a ele associadas. Acredita-se que o ProFap-R possa também ser aplicado na reengenharia de sistemas de grande porte. Para a obtenção de melhores resultados com o uso do ProFap-R recomenda-se utilizar um *framework* no domínio do sistema legado cuja arquitetura seja adequada para a reengenharia. São sugeridos os seguintes trabalhos futuros: i) descrever estratégias de teste de regressão e associá-las à etapa "Efetuar Teste de Integração e Aceitação"; ii) explicitar no processo ProFap-R outras ferramentas computacionais que podem ser utilizadas para apoiar cada etapa; e iii) conduzir estudos de casos de reengenharia com sistemas de grande porte para averiguar a eficiência do ProFap-R nesses casos.

Referências

Cagnin, M. I., Maldonado, J., Germano, F., Penteado, R. (2003). PARFAIT: Towards a Framework-based Agile Reeng. Process. In: Agile Develop. Conf., Salt Lake City, Utha.

Cagnin, M. I., Acosta, A., Gonçalves, C., Vieira, C., Blanes, D., Smaka, E., Sandim, H., Luna, J., Costa, K., Alvarenga, M., Silva, M., Oliveira, M. (2012). Gestão de Inf. das Famílias Beneficiadas no Prog. da Saúde da Família no Município de Campo Grande-MS. Proj. Pesq. Aprovado no Edital FUNDECT/DECIT-MS/CNPq/SES N° 04/2012.

Cagnin, M. I., Turine, M., Silva, M., Landre, G., Oliveira, L., Lima, V., Santos, M., Paiva, D., Carromeu, C. (2013). ProFap: Processo Colaborativo de Manutenção de Software. In: X Workshop de Manutenção de Softw. Moderna (WMSWM'2013), Salvador, Bahia.

Carromeu, C., Paiva, D. M. B., Cagnin, M. I., Rubinsztejn, H. K. S., Turine, M. A. S., Breitman, K. (2010). Component-Based Architecture for e-Gov Web Systems Development. In: 17th IEEE International Conf. and Workshops on Eng. of Computer-Based Systems, Oxford, UK.

Chikofsky, J. E., Cross, J. H. (1990). Reverse engineering and design recovery: A taxonomy. IEEE Software, v. 7, n. 1, p. 13-17.

Fayad, M. E.; Johnson, R. E. (2000). Domain-specific application frameworks: Frameworks experience by industry. John Wiley & Sons, 1ª ed.

Guido, A.L., Paiano, R., Pandurino, A., Mainetti, L. Transforming legacy systems into user-centred web applications (2010). Management of the Interconnected World - ItAIS: The Italian Association for Information Systems, p. 395-401.

Pérez-Castillo, R., de Guzmán, I., Caballero, I., Piattini, M. (2013). Software modernization by recovering Web services from legacy databases. Journal of Software: Evolution and Process, v. 25, n. 5, p. 507-533.

Pressman, R. S. (2011). Engenharia de Software: Uma Abordagem Profissional. McGraw Hill, 7ª ed.

Rodríguez-Echeverría, R., Conejero, J., Clemente, P., Preciado, J., Sánchez-Figueroa, F. (2011). Modernization of legacy web applications into rich internet applications. In: 7th Model-Driven Web Engineering Workshop, 7059 LNCS, 236-250, Paphos, Cyprus.

Ruiz, M., Espanã, S., Gonzalez, A. (2012). Model-driven organisational reengineering: A framework to support organisational improvement. In: 38th Latin America Conference on Informatics, Medellín-Colômbia.

Sommerville, I. (2011). Engenharia de Software. Pearson Education Brasil, 9ª ed.