Manutenção e Evolução de um Gerador de Aplicações Desenvolvido com Linha de Produtos de Software

Simone de S. Borges, Tatiane T. Ferreira, Rosângela A. D. Penteado

DC/UFSCar – Departamento de Computação
Universidade Federal de São Carlos
Caixa Postal 676 – 13565-905 – São Carlos – SP – Brasil

{simone_borges,rosangel}@dc.ufscar.br, tatiane_tosta@comp.ufscar.br

Abstract. *A Web application generator created from Software Product Line for the rehabilitation clinics management systems domain was used for the generation of systems in a related domain. Some tasks of corrective and preventive maintenance were necessary so that the system belonging to the related domain could be generated. This work describes how those activities were accomplished in the generator and/or in the artefacts used in its construction, as well as the creation of a software configuration management process for that generator, to control the generation of such applications.*

Resumo. *Um gerador de aplicações para a Web criado a partir de Linha de Produtos de Software, para o domínio de clinicas de reabilitação física, foi utilizado para a geração de sistemas em um domínio conexo a esse. Algumas tarefas de manutenção corretiva e preventiva foram necessárias para que o sistema pertencente ao domínio conexo pudesse ser gerado. Este trabalho comenta como essas atividades foram realizadas no gerador e/ou nos artefatos usados em sua construção, bem como a criação de um processo de Gestão de Configuração de Software para esse gerador, para controlar sa diferentes versões que podem ser geradas de tais aplicações.*

1. Introdução

Linha de Produtos de Software (LPS) consiste em uma família de sistemas de software que possuem um núcleo de funções em comum, cuja vantagem é formar um conjunto de recursos reusáveis e funções variáveis, que permitem a especialização das aplicações para atender a domínios específicos [Pohl *et al*., 2005] [Gomaa, 2007]. Uma forma de implementar uma LPS e automatizar parte do processo de desenvolvimento de software é por meio de geradores de aplicações. Esses geradores aceitam uma especificação, validam-na e geram seus artefatos. Uma Linguagem de Modelagem da Aplicação (LMA) representa as abstrações (modelos) das aplicações [Weiss e Lai, 1999] e é uma das formas de documentar essas especificações.

GAwCRe [Pazin, 2004] é um gerador que possibilita o desenvolvimento de sistemas para a Web, pertencentes ao domínio de clínicas de reabilitação física. Foi desenvolvido com base em técnicas de reúso, linguagem de padrões SiGCli [Pazin *et al.*, 2004] e LPS [Pazin, 2004]. A utilização desse gerador, para o desenvolvimento de aplicações pertencentes a domínios conexos ao seu, demandou diversas modificações executadas *ad hoc* [Freitas, 2006]. Quando da utilização desse gerador, para avaliar tais

modificações, observou-se que eram necessárias atividades de manutenção perfectiva e corretiva para que a SiGCLi fosse preservada ao máximo.

Tarefas de manutenção são processos dinâmicos, em que sistemas de software passam por modificações constantes, sendo recomendável a adoção de um processo de Gerência de Configuração de Software (GCS) para controlar e rastrear essas modificações de forma disciplinada [Yu e Ramaswamy, 2006]. Assim, este artigo tem por objetivo discutir um processo de GCS adotado para o gerador GAwCRe para apoiar sua manutenção e evolução.

A organização deste artigo é a seguinte: na Seção 2 o GAwCRe é apresentado; na Seção 3 algumas das atividades de manutenção realizadas são discutidas; alguns trabalhos correlatos são apresentados na Seção 4 e, na Seção 5 são feitas considerações sobre os resultados obtidos, armadilhas encontradas durante o processo de manutenção e sugestões para a realização de trabalhos futuros.

2. GAwCRe – Gerador de Aplicações baseadas na Web para Clínicas de Reabilitação

GAwCRe é um gerador de aplicações para a Web desenvolvido com base em Linha de Produtos de Software, no domínio de clínicas de reabilitação física (Fisioterapia, Terapia Ocupacional e Educação Física). A instanciação do gerador é feita a partir de uma LMA definida com base na linguagem de padrões SiGCli. Para armazenar as informações referentes à linguagem de padrões e à LMA, um meta-modelo em XML foi elaborado contendo um conjunto de *tags* e atributos de *tags*, que são utilizadas para compor os artefatos da aplicação. Na inicialização do GAwCRe, o meta-modelo é lido e com as informações ali contidas, a interface do gerador é definida e os padrões disponíveis para a geração das aplicações são apresentados. O total de aplicações distintas que podem ser geradas pelo GAwCRe, utilizando a SiGCli como foi originalmente criado é de seiscentas e oitenta e oito aplicações. Considerando a flexibilidade da linguagem XML, outras linguagens de padrões podem ser mapeadas no meta-modelo, que viabiliza o reúso do gerador em outros domínios [Pazin, 2004].

Originalmente o GAwCRe utilizava o SGBD Oracle [Oracle, 2008]. Após um processo de manutenção adaptativa *ad hoc* passou a utilizar o SGBD MySQL [Rizzo, 2005]. Por um outro processo *ad hoc* [Freitas, 2006], foi proposta a utilização do gerador GAwCRe como ferramenta para a geração automatizada de artefatos em uma abordagem de reengenharia ágil - ARA [Cagnin, 2005]. Devido à dificuldade em encontrar um sistema exemplo pertencente ao seu domínio, foi realizada a reengenharia em um sistema pertencente a um domínio conexo. Assim, foi necessária a expansão do GAwCRe para atendê-lo. Os seguintes problemas foram encontrados: a) o código não segue padrão de codificação e não é flexível; b) há erros de lógica e essa é complexa de ser assimilada; c) faltam instruções para a instalação e operação do gerador; d) o registro e a validação das manutenções anteriores é superficial ou inexistente; e) a instalação dos artefatos gerados não é automatizada e faltam instruções para realizá-la; f) há presença de *bad smells* [Fowler, 2004] (código duplicado, métodos extensos, etc). Assim, novas atividades de manutenção foram necessárias como relatado a seguir.

3. Atividades de Manutenção realizadas no GAwCRe

Para que o gerador pudesse ser utilizado para a produção de sistemas pertencentes a domínios conexos àquele para o qual foi construído, algumas atividades de manutenção corretiva foram realizadas e um processo de gestão de configuração de software foi definido (Seção 3.1). Atividades de manutenção perfectiva ou evolutiva foram realizadas somente no código XML existente e não na linguagem de padrões SiGCli, devido à facilidade existente com esse tipo de implementação comentadas nas Seções 3.2 e 3.3.

3.1 Manutenção Corretiva

Para controlar as modificações realizadas, de modo a garantir a integridade do gerador e evitar a introdução inadvertida de problemas, o primeiro passo foi utilizar um processo de GCS Tradicional [Borges, 2007], levando em consideração propostas de outros modelos encontrados [Hass, 2002] [Yu e Ramaswamy, 2006] [SEI, 2008] [IEEE 828, 2005]. As atividades essenciais que devem ser realizadas são: versionamento, *merge* básico, uso de ramificação ou *branching*, documentação colaborativa, controle de mudanças (registro de defeitos, pedidos de melhoria e andamento das tarefas do projeto) e uso de interfaces para manutenção e realização de tarefas. A definição, instalação e a configuração das ferramentas de apoio foram realizadas considerando a estabilidade, a adequação ao projeto e que fossem de uso livre. Assim, optou-se pelo TortoiseSVN [CollabNet, 2008] – para controle de versões; Trac [Edgewall, 2008] – para o controle de mudanças; a suíte MySQL Enterprise Server 5 [MySQL, 2008]; Eclipse [Eclipse, 2008]; JUnit [JUnit, 2008].

Na estratégia de GCS definida para as tarefas de Controle de Versões (CV) e Controle de Mudanças (CM), os principais itens de configuração foram separados por grupos [Yu e Ramaswamy, 2006]. O grupo denominado *Producao* inclui o conjunto de classes responsáveis pela geração das aplicações e da interface de instanciação do GAwCRe. Nesse grupo também se encontra o pacote *pl*, que contém as classes que implementam o padrão *Persistent Layer* [Yoder *et al*., 1998]. No grupo *LingPadrao* estão os meta-modelos mapeados em linguagem XML (mapeamento original da SiGCli e outros arquivos XML, variantes do original, e que foram adaptados para domínios conexos). Por último, o grupo *Produto* que engloba todos os artefatos instanciados a partir do gerador, sendo que cada aplicação tem sua própria linha de desenvolvimento, não misturando os artefatos. Foram criados três repositórios distintos para cada grupo, todos seguem o modelo *trunk*, *tag*, *branch* como uma forma de padronização. A Figura 1 ilustra parte do processo utilizado.

Uma vez estabelecida a estratégia de CV e de CM, todos os pedidos de manutenção foram registrados na ferramenta de controle de mudanças e foram descritos e categorizados como melhoria, defeito ou tarefa. Para o problema do método utilizado para o carregamento do arquivo XML que foi implementado sem flexibilidade, a solução foi criar uma interface que precede a inicialização do gerador e permite a localização e a escolha do meta-modelo XML desejado.

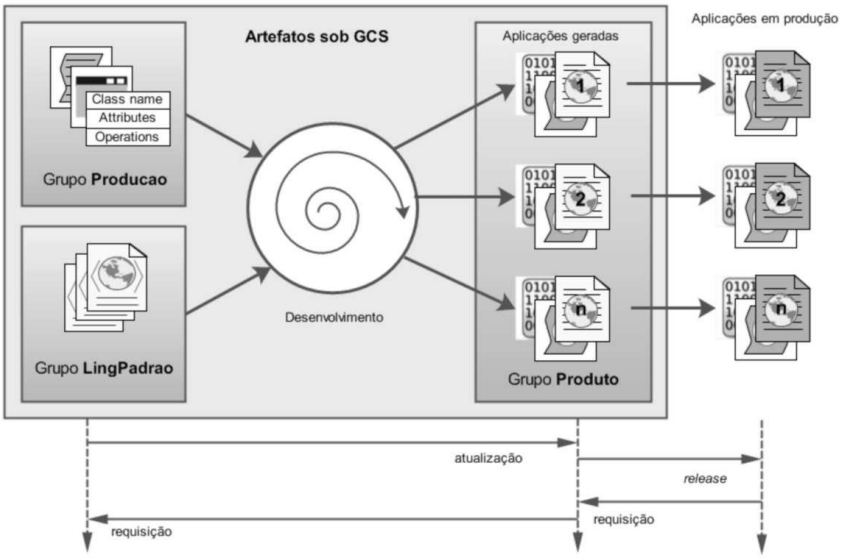

Figura 1. Artefatos sob versionamento, os itens de configuração são divididos em grupos de acordo com sua função em relação à arquitetura do gerador. (Adaptado de [Yu e Ramaswamy 2006]).

Nos *scripts* SQL gerados pelo GAwCRe, o nome da base de dados das aplicações geradas é sempre o mesmo. Dessa forma, a cada nova aplicação gerada, o banco de dados necessita ser "limpo" para que novas tabelas sejam criadas, perdendo as tabelas da aplicação anterior. Como muitas classes responsáveis pela criação e manutenção da base de dados estão inter-relacionadas, se as alterações forem feitas sem que haja controle efetivo sobre elas, pode-se incorrer em problemas futuros. Assim, testes automatizados, que exercitam a camada de persistência do gerador, estão sendo criados a fim de detectar a introdução de problemas.

3.2. Manutenção Perfectiva

A manutenção perfectiva foi utilizada para expandir o domínio do GAwCRe para clínicas de domínios conexos (odontológica, psicológica, clínicas médica, etc.) sem entretanto perder o domínio de clínicas de reabilitação e seguindo as atividades previstas na reengenharia ágil [Freitas, 2006]. Entre os sistemas testados, dois são apresentados a seguir e algumas considerações são feitas.

3.2.1 Sistema OdontoPlus

Para alterar a SiGCli expandindo seu domínio para clínicas odontológicas [Ferreira, 2007], mas preservando também a funcionalidade do domínio de clínicas de reabilitação, utilizou-se um sistema denominado OdontoPlus, obtido via Internet [NetSource, 2008]. Sua funcionalidade permite o controle completo sobre pacientes, consultas e a parte financeira do consultório. A utilização do GawCRe para gerar essa aplicação apresentou alguns problemas, pois os padrões SiGCli não cobrem totalmente o domínio dessa aplicação, sendo necessárias intervenções. Ao utilizar o primeiro padrão da SiGCli que trata da "Alocação de Recursos", notou-se que nem todas as classes consideradas obrigatórias eram necessárias para esse sistema. O mesmo ocorre com os atributos dessas classes: alguns não são necessários para a aplicação e outros que são

necessários, mas que não estão representados na SiGCli. Algumas das funções existentes no sistema que está passando por reengenharia não são cobertas pelos padrões existentes na SiGCli.

Para superar essas diferenças algumas alternativas foram elaboradas como soluções. Com relação à diferença dos atributos nas classes do sistema alvo para o legado (OdontoPlus) a solução mais fácil, mas não viável, é remover os atributos não necessários para esse domínio e incluir outros necessários nas classes da SiGCli. Essa solução geraria uma aplicação com os atributos iguais aos do sistema OdontoPlus, mas o domínio da SiGCli seria alterado o que acarretaria a não geração de aplicações para clínicas de reabilitação. Outra solução, a melhor até o momento, é não remover das classes da SiGCli os atributos não necessários para clínicas odontológicas e inserir apenas aqueles atributos que não afetam o domínio original. Dessa maneira, a aplicação gerada tem classes com atributos de ambos os domínios e as alterações ainda necessárias nos atributos das classes teriam que ser feitas no próprio sistema legado. Para resolver o problema das classes ausentes sem alterar o domínio da SiGCli optou-se por modificar somente as variantes dos padrões e criar novas variantes para incluir tais classes. No entanto, as classes obrigatórias da SiGCli que não são necessárias para o sistema de clínicas odontológicas continuariam a existir, pois se modificadas ou retiradas o domínio da SiGCli é afetado. Com isso, o sistema alvo gerado a partir da SiGCli modificada, passou a ter as classes necessárias para clinicas odontológicas e outras não utilizadas por essa aplicação que foram criadas por serem obrigatórias na SiGCli.

Algumas funções do sistema legado, como gráfico de movimento, backup do banco de dados, relatórios de faturamento e controle de usuários do sistema existentes no OdontoPlus, não podem ser incluídas na SiGCli, pois ela não apóia esse tipo de função. A cada modificação da SiGCli novos artefatos são gerados e comparados com o sistema legado. Como comentado anteriormente a cada aplicação perde-se todo o banco de dados da aplicação anterior, assim foi possível apenas salvar os *scripts* da criação das tabelas de cada aplicação gerada. Mesmo tendo sido testadas, somente com as soluções citadas anteriormente, não foi possível expandir o domínio da SiGCli para clínicas odontológicas, pois, da maneira como os seus padrões estão organizados não são permitidas muitas alterações. Para isso é necessário o estudo dos domínios e a reestruturação da SiGCli, com a inclusão de novos padrões e possível alteração nos já existentes. Durante o processo de alteração da SiGCli não foi utilizado nenhum controle de versão sobre os artefatos gerados nem sobre o meta-modelo no qual está mapeada a SiGCli, esses apenas foram nomeados de forma usual, por data e detalhes de seu conteúdo.

3.2.2 Psychologist

O sistema Psychologist atende clínicas de psicologia [Biomanager, 2008]. Sua implementação utilizando o GAwCRe ocorreu sem intervenções no código fonte dos artefatos gerados, alterando-se apenas o meta-modelo mapeado em XML. Desse modo, somente os cadastros realizados no sistema legado Psychologist foram obtidos, acrescentando-se ou retirando-se atributos no meta-modelo e gerando uma nova aplicação.

As mesmas dificuldades descritas na seção anterior ocorreram na reengenharia desse sistema, também constatou-se que somente alterando o meta-modelo é impossível redefinir a apresentação dos sistemas gerados. Isso ocorre porque os artefatos gerados pelo GAwCRe utilizam folhas de estilo em cascata (CSS) a fim de customizar o "visual" das aplicações. No GAwCRe foram empregadas três formas de declaração de *tags* CSS (*inline*, *embedded* e *external*) resultando em características indesejáveis e comportamento instável, dificultando a manutenção, pois o código CSS encontra-se espalhado. Parte do problema foi solucionado adotando-se somente a implementação do tipo *external*, criando um único arquivo que contém o estilo da aplicação, o qual é criado com o mesmo nome do artefato gerado, tornando-o único. Resta ainda eliminar o código CSS nos *templates* e no meta-modelo, concentrando-o no arquivo externo único citado. Tais alterações facilitam futuras atividades de manutenção, pois os arquivos relacionados à apresentação dos elementos visuais encontram-se agora definidos em um único arquivo.

4. Trabalhos Correlatos

Dentre os trabalhos relacionados ao desenvolvimento e a evolução de LPS [Wijnstra, 2004] [Loesch e Ploedereder, 2007] [Metzger e Pohl, 2007] [Gomaa e Shin, 2007], poucos abordam GCS para Linha de Produtos e entre os que tratam o assunto, a maior parte enfoca a gerência de componentes ou frameworks de software, e nenhum aborda especificamente geradores de aplicação.

Yu e Ramaswamy (2006) desenvolveram pesquisa semelhante a esta, dividindo os itens de configuração em fundamentais, customizáveis e artefatos gerados. Os autores evidenciam os benefícios obtidos pela implementação de uma LPS, relatam as dificuldades encontradas na evolução desses sistemas e propõem soluções como versionamento e controle de mudanças, enfatizando a necessidade de controle principalmente sobre as partes customizáveis, consideradas o ponto crítico na manutenção e evolução desses sistemas.

5. Considerações Finais e Trabalhos Futuros

A manutenção de sistemas legados não é trivial, bem como a de sistemas desenvolvidos sem que as atividades de engenharia de software tenham sido realizadas. Dois sistemas legados, Psychologist e OdontoPlus, foram utilizados a fim de se verificar a possibilidade de empregar o GAwCRe na reengenharia de sistemas que possuíssem as mesmas funções dos legados. Entre os problemas encontrados estão a incompatibilidade de classes, de atributos e de métodos existentes na linguagem de padrões SiGCli e os sistemas legados utilizados; a falta de mecanismo para controle de versões no gerador; a base de dados das aplicações geradas sempre são armazenadas com o mesmo nome, que inviabiliza o armazenamento de múltiplas versões de um sistema ao mesmo tempo; a codificação em CSS, responsável pela apresentação dos artefatos gerados, que não estava padronizada. Assim, o GAwCRe passou por manutenção perfectiva para que sistemas pertencentes a domínios conexos ao seu, também possam ser gerados. Tarefas de manutenção corretiva também foram realizadas para que fosse possível a utilização do gerador nessas atividades.

Durante a manutenção perfectiva notou-se a possibilidade de estender a linguagem de padrões originalmente construída, além de adaptar métodos e atributos relacionados a aplicações específicas. Isso pode ser resolvido com a escolha, logo ao iniciar a geração, do tipo de sistema que será produzido com o GAwCRe: se de clínicas de reabilitação ou de clínicas médicas, odontológica ou psicológica. Observou-se que a falta de uma gestão de controle de configuração dificultava o uso efetivo do GAwCRe, sua manutenção e também a geração de artefatos. A adoção de um processo de GCS (controle de versões e controle de mudanças) possibilitou que a manutenção realizada no sistema Psychologist, ocorresse em ambiente controlado e com a documentação de todos os passos relevantes. Assim, agregou-se rastreabilidade ao processo de manutenção e aprimorou-se a sua manutenibilidade. Com o sistema OdontoPlus, o controle dos arquivos foi feito manualmente, conseqüentemente sua rastreabilidade é menor.

Um benefício obtido com essa abordagem é a preservação (*trunk*) tanto do GAwCRe quanto da linguagem de padrões SiGLi e quaisquer adaptações para domínios conexos podem ser criadas (*branch*), o que permite a exploração da variabilidade do gerador. A proposta de Freitas (2006) prevê alterações unicamente no meta-modelo para se equiparar o sistema alvo ao legado. Todavia, caso sejam solicitadas, essas alterações, podem ser realizadas, sucessivas vezes, no sistema alvo a fim de se obter essa equiparação, e no caso do artefato necessitar ser gerado novamente, as alterações anteriores não são perdidas e podem ser incorporadas à nova versão.

Um próximo passo, será verificar a possibilidade de expansão do domínio coberto pelos padrões da SiGCli, com o remanejamento dos padrões existentes e/ou a adição de novos padrões que permitam a geração de aplicações para domínios conexos com maior flexibilidade e facilidade. O processo de GCS definido pode continuar a ser utilizado, pois sua proposta é apoiar a evolução e manutenção do gerador e da linguagem de padrões.

Referências

Biomanager. (2008). Psychologist Software. http://www.biomanager.com.br/PsicoSoft/. (Acesso em 09 de março de 2008).

Borges, S. S. (2007). *Um Estudo de Qualidade na Manutenção de Gerador de Aplicações*. Monografia de exame de qualificação de mestrado apresentada ao PPG-CC, DC-UFSCar.

Cagnin, M. I. (2005) *PARFAIT: Uma contribuição para a reengenharia de software baseada em linguagens de padrões e frameworks*. Tese de Doutorado, ICMC/USP.

Eclipse. (2008). Eclipse Foundation. Disponível em http://www.eclipse.org/ (Acessado em 08 de fevereiro de 2008).

Edgewall Software. (2008). Trac – Integrated Software Configuration Management & Project Management. Disponível em http://trac.edgewall.org/. (Acesso em 18 de fevereiro de 2008).

Ferreira, T. T. (2007). Avaliação de um Processo de Manutenção Usando Gerador de *Aplicações*. Projeto de Iniciação Científica. PIBIC/CNPq-UFSCar.

Fowler. M. (1999). *Refactoring: Improving the Design of Existing Code*. Addison Wesley.

Freitas, R.G. (2006). *Utilização de Geradores de Aplicação em Processos Ágeis de Reengenharia*. Dissertação de Mestrado – PPG-CC, DC-UFSCar.

Gomaa, H. Shin, Michael E. (2007). *Automated Software Product Line Engineering and Product Derivation*. Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07).

Hass, A. M. J. (2002). *Configuration Management Principles and Practice*. Addison Wesley.

IEEE Standard 828 (2005). *IEEE Standard for Software Configuration Management Plans*. In: IEEE Software Engineering Standard Collection.

JUnit (2008). *Resources for Test Driven Development*. Disponível em http://www.junit.org/. (Acesso em 06 de março de 2008).

Loesch, F. Ploedereder, E. (2007). *Optimization of Variability in Software Product Lines*. In Proceedings of the 11th international Software Product Line Conference (September 10 - 14, 2007). International Conference on Software Product Line.

Metzger, A. Pohl, K. (2007). *Variability Management in Software Product Line Engineering*. ICSE Companion, pp. 186-187, IEEE Computer Society.

MySQL AB. (2008). MySQL Enterprise Server 5.0. Disponível em http://www.mysql.com/products/enterprise/server.html. (Acesso em 08 de fevereiro de 2008).

NetSource. (2008). OdontoPlus. Disponível em http://www.netsource.com.br/. (Acesso em 09 de março de 2008).

Oracle. (2008). Oracle. Disponível em http://www.oracle.com/global/br/index.html (Acesso em 09 de março de 2008).

Pazin, A. (2004). *Um Gerador de Aplicações para o Domínio de Clínicas de Reabilitação*. Dissertação de Mestrado – PPG-CC, DC-UFSCar.

Pazin, A.; Penteado, R.; Masiero, P.C. (2004). *SiGCli: A Pattern Language for Rehabilitation Clinics Management*. In: SugarLoafPlop' 2004, IV Conferência Latino-Americana em Linguagens de Padrões para Programação, Porto das Dunas – CE, p. 1-25.

Pohl, K., Böckle, G., van der Linden, F. (2005). *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer.

Rizzo, J. A. M. (2005). *Validação de Processos Ágeis de Reengenharia e de Geradores de Aplicações.* Relatório de Iniciação Científica. PIBIC/CNPq-UFSCar.

SEI - Software Engineering Institute. (2008).. *Software Product Lines*. Carnegie Mellon University. Disponível em http://www.sei.cmu.edu/productlines/ (Acesso em 20 de fevereiro de 2008).

Weiss, D.; Lai, C. T. R. (1999). *Software Product-Line Engineering: a family-based software development process*. Ed. Addison Wesley.

Wijnstra, J. G. (2003). *Evolving a Product Family in a Changing Context*. Proceedings of the Fifth International Workshop on Product Family Engineering (PFE-5), LNCS 3014, Springer Verlag.

Yoder, J.W.; Johnson, R.E.; Wilson, Q.D. (1998). *Connecting Business Objects to Relational Databases.* In: Conference on the Pattern Languages of Programs, 5, Monticello-IL, EUA. Proceedings.

Yu, L. Ramaswamy, S. (2006). *A Configuration Management Model for Software Product Lines*. INFOCOMP Journal of Computer Science, pp 1-8.