SACRES 2.0: Suporte à Recomendação de Configurações considerando Múltiplos *Stakeholders*

Lucas Lazzari Tomasi[1], **Jacob Stein**[1], **Ingrid Nunes**[1]

[1]Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, RS, Brasil

{lltomasi,jacob.stein,ingridnunes}@inf.ufrgs.br

Abstract. *Feature models (FMs) are used to represent and organize the variability in a software product line, and the task of configuring a FM in order to generate a valid product configuration thus becomes a key activity. This task, which is known to be hard and time consuming, gets even harder when many stakeholders are involved in the process of configuration. The tool presented in this paper has the purpose of supporting the configuration process with multiple stakeholders, recommending optimal configurations based on their preferences.*

Resumo. *Feature Models (FMs) são usados para representar e organizar a variabilidade em uma linha de produto de software, e a tarefa de configurar um FM de modo a gerar a uma configuração de produto válida torna-se, portanto, uma atividade chave. Essa tarefa, conhecida por ser difícil e demorada, torna-se ainda mais complicada quando múltiplos stakeholders estão envolvidos no processo de configuração. A ferramenta apresentada neste artigo tem o propósito de dar suporte ao processo de configuração com múltiplos stakeholders, recomendando configurações ótimas baseadas em suas preferências.*
URL do video: `http://www.youtube.com/watch?v=HeHT8JXlgR4`.

1. Introdução

O reuso de componentes não é um conceito novo, é uma das estratégias usadas para a redução de custos e esforço no desenvolvimento de sistemas de software. As linhas de produto surgiram com esta motivação, sistematizando o reuso, criando famílias de produtos com características comuns entre si, permitindo um aumento de produção, mas com a possibilidade de customização. Na indústria de software a combinação dos conceitos de famílias de produtos e customização em massa deram origem às chamadas de Linhas de Produto de Software (LPS) [Clements and Northrop 2002]. Segundo Linden e Pohl (2005), LPS referem-se às técnicas de engenharia para a criação de sistemas de software similares a partir de um conjunto compartilhado de partes, através de uma forma sistemática de construção de aplicações. Por meio da organização e sistematização do reuso é possível diminuir custos de desenvolvimento, melhorar a qualidade do software e reduzir o tempo de produção.

Nas LPS, o uso de *feature models* (modelos de *features*) destaca-se como uma forma de representar e gerenciar a variabilidade dentro de uma LPS. Essa variabilidade é expressa por meio de *features*. Uma *feature* representa uma característica ou propriedade do sistema que é relevante para algum *stakeholder* [Czarnecki and Eisenecker 2000]. O

Feature Model (FM) é uma representação gráfica em forma de árvore que permite visualizar as features disponíveis de serem selecionadas de forma a criar um novo produto customizado [Kang et al. 1990]. O FM modela as propriedades comuns e variáveis dos produtos possíveis de uma LPS, incluindo suas interdependências. Também é possível a definição de restrições entre certas combinações de *features*, quando for necessário que duas sejam mutualmente exclusivas, por exemplo.

A partir de um FM definido é possível a criação de configurações de produtos. Uma configuração de FM (i.e. configuração de um produto) é a seleção de um conjunto de *features* contidas nesse modelo. Quando esta seleção de *features* respeita as restrições presentes no modelo ela é chamada de configuração válida. Dado que FMs podem ter um grande número de *features* disponíveis, temos um número exponencial de configurações possíveis. Desta forma, escolher um conjunto válido de *features*, satisfazendo as restrições do FM e as preferências de *múltiplos* stakeholders torna-se uma tarefa nada trivial, exigindo suporte computacional [Mendonca and Cowan 2010].

Quando há um número maior de indivíduos envolvidos no processo de configuração de um mesmo produto torna-se complicado de encontrar um consenso para determinar quais *features* devem estar presentes no produto final, de forma que os participantes do processo fiquem suficientemente satisfeitos. É perceptível a necessidade de uma ferramenta para dar suporte ao processo e aos participantes. Portanto, apresentamos neste artigo o SACRES 2.0, um plugin para a IDE Eclipse que implementa uma nova abordagem [Stein et al. 2014] para dar suporte à configuração de FMs com base nas preferências de múltiplos *stakeholders*.

Este artigo está organizado da seguinte forma. A Seção 2 descreve o problema, exemplificando os tipos de conflitos endereçados em nossa ferramenta. A Seção 3 apresenta detalhadamente a ferramenta desenvolvida e um pequeno exemplo explicativo. A Seção 4 discute brevemente alguns trabalhos relacionados e, finalmente, a conclusão é feita na Seção 5.

2. O Problema

Para chegar na definição de uma configuração de um produto é comum termos opiniões de múltiplas partes envolvidas no processo. Nesta abordagem consideramos um cenário onde vários *stakeholders* envolvidos no processo de configuração de um produto podem expressar suas opiniões e preferências. *Stakeholders* podem ser qualquer indivíduo que tem um interesse no domínio, como por exemplo: gerentes técnicos e de marketing, programadores, usuários finais e clientes. Considerando esses diversos indivíduos, que possuem necessidades, interesses e preocupações distintas com relação ao produto a ser desenvolvido, podemos perceber que conflitos devam surgir devido às preferências divergentes entre os *stakeholders*.

A participação de mais de um *stakeholder* nesse processo o torna mais complexo, pois podem existir preferências conflitantes entre eles, de modo que seja impossível recomendar uma única configuração que satisfaça-os razoavelmente. Isso acontece, por exemplo, quando dois *stakeholders* expressam suas preferências sobre uma mesma *feature* de modo que um deles deseja que ela esteja presente no produto, porém o outro prefere que ela não esteja. Um segundo tipo de conflito acontece nos grupos de *features* alternativas, onde somente uma das *features* do grupo pode ser selecionada. No momento

que um *stakeholder* selecionar F1, ele automaticamente impede que outro *stakeholder* selecione F2, considerando que as duas (F1 e F2) estão em um mesmo grupo. Um terceiro tipo de conflito surge com relação às restrições de integridade do FM onde, caso um primeiro *stakeholder* selecione a *feature* F1 que requer a seleção de F2, ele automaticamente faz com que as ambas sejam selecionadas, podendo causar um futuro conflito caso algum stakeholder não queira uma delas no produto final.

Considerando os conflitos descritos, percebe-se a necessidade de uma abordagem para endereçá-los. Desta forma, a presente ferramenta implementa uma abordagem [Stein et al. 2014] que nós desenvolvemos a qual propõe um metamodelo que permite a criação de configurações de *stakeholder*. Cada configuração de stakeholder representa as preferências estabelecidas por um indivíduo sobre as features do FM. Com base nessas configurações buscamos uma configuração final que satisfaça da melhor maneira os *stakeholders* envolvidos. Assim, na próxima seção mostraremos as principais funcionalidades e características da ferramenta SACRES 2.0.

3. Funcionalidades e Arquitetura da Ferramenta

O SACRES 2.0 é um plugin para a IDE Eclipse desenvolvido em Java que funciona de maneira integrada com outro plugin chamado FeatureIDE[1]. Dentre as possíveis ferramentas existentes para a integração com o SACRES 2.0, o FeatureIDE foi o escolhido por sua fácil usabilidade e por possuir cobertura de todo o processo de desenvolvimento de uma LPS, incluindo assim, funcionalidades desejadas para dar suporte à nossa implementação. Além disso, ele dispõe de uma documentação bem detalhada e está em constante desenvolvimento. O FeatureIDE objetiva a redução de esforços para a construção de ferramentas para novas e existentes abordagens de LPS. O plugin também possui diversas funcionalidades pertinentes para a aplicação da nossa ferramenta, entre as principais citamos: criação e edição de modelos de *features*, análise automática do modelo, criação e edição de configurações de produtos. Ele utiliza *views* específicas que permitem a visualização com detalhes dos FMs e das possíveis configurações.

A abordagem acima mencionada, implementada pelo SACRES 2.0, foi anteriormente implementada na forma de um *protótipo*, para viabilizar um experimento para avaliar diferentes estratégias da teoria da escolha social, que foram instanciadas para o contexto de FMs [Stein et al. 2014]. Este protótipo, referenciado como SACRES 1.0, carece das funcionalidades relacionadas à criação e edição de FMs disponíveis no FeatureIDE, pois utiliza um banco de dados para armazenamento dos FMs previamente definidos. Sua interface permite apenas a criação e edição de configurações de *stakeholders* e grupos de SHs. Assim sendo, nossa ferramenta tem como propósito a integração das funcionalidades do FeatureIDE com a recomendação de configurações ótimas envolvendo múltiplos *stakeholders* realizada pelo SACRES 1.0. Além disso, a recomendação de configurações é realizada de forma mais otimizada.

A Figura 1 mostra a arquitetura do plugin desenvolvido com a interação entre as partes envolvidas. Nela, podemos ver que o FeatureIDE é o responsável pela criação e edição do modelo de *features* e pela visualização das configurações recomendadas. O SACRES 2.0 contém as classes com suporte ao metamodelo e a implementação dos algoritmos de recomendação para as estratégias de escolha social usadas. Além disso, o plugin

[1] `http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/`

possibilita a criação e edição de configurações que são salvas em arquivos individuais e organizadas em pastas para cada grupo de *stakeholders* associando-os a um modelo de *features*. Desse modo, ela também é responsável pelo gerenciamento dos arquivos, desde os novos arquivos com as configurações de stakeholder baseadas no metamodelo até os resultados das recomendações que podem ser visualizados no FeatureIDE. O SACRES 2.0 faz o uso do CSP Solver CHOCO[2] de forma a eliminar de maneira mais eficiente as configurações consideradas inválidas pelas restrições duras, gerando o que chamamos de *conjunto de consideração*, utilizado pelas estratégias de escolha social para buscar as configurações ótimas.

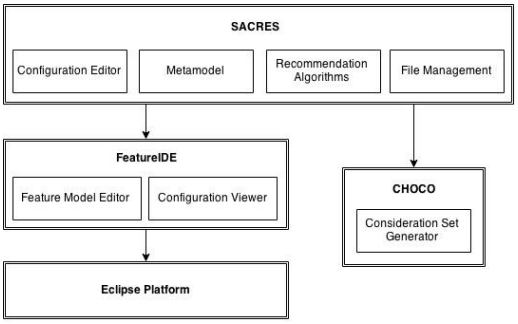

Figura 1. Arquitetura da ferramenta.

Para uma melhor compreensão do funcionamento da ferramenta iremos detalhar suas funcionalidades através de um pequeno exemplo. Para isso, criamos um FM simples e associamos um grupo de três *stakeholders* a esse modelo.

3.1. Metamodelo

Na solução proposta por Stein (2014), na qual o plugin se baseia, foi definido um metamodelo para melhorar a expressão de preferências dos *stakeholders*, flexibilizando o processo, através de definições de preferências de *features* de maneira quantitativa, representando o nível de preferência sobre as *features*. Em outras palavras, os *stakeholders* definem um número no intervalo [-1,1] que expressa o quanto eles querem que uma *feature* esteja presente na configuração (representado por valores positivos) ou o quanto não desejam que uma *feature* esteja na configuração final (representado por valores negativos). Essas preferências são chamadas de restrições do tipo *soft* ou brandas.

Além das restrições brandas o metamodelo também permite que sejam estabelecidas restrições do tipo *hard* ou duras, que por sua vez representam a obrigatoriedade de uma *feature* estar presente na configuração, quando utilizada uma restrição *hard* positiva, ou de não estar presente, quando utilizada uma restrição *hard* negativa. Dessa forma podemos dizer que uma configuração de um *stakeholder* é composta por um conjunto de restrições *hard* positivas e negativas e por um conjunto de restrições *soft* positivas e negativas associados a um modelo de *features*. Para medirmos o nível de satisfação de um *stakeholder* em relação à uma configuração usaremos as restrições do tipo *soft*. A soma das restrições brandas satisfeitas em uma configuração nos dá um valor numérico, chamado de satisfação de *stakeholder*, e é esse valor que desejamos maximizar. Já as restrições duras servem para eliminar possíveis configurações conflitantes em uma primeira etapa, diminuindo o espaço de busca a ser considerado pela segunda etapa.

[2] `http://choco-solver.org/?q=Choco3`

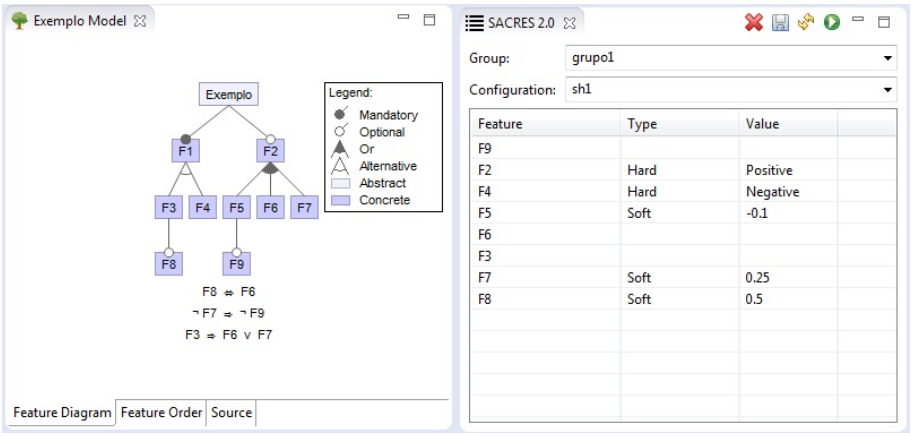

Figura 2. *Feature Model* e possível configuração de *stakeholder*.

A Figura 2 apresenta o FM criado através do FeatureIDE como exemplo e a *view* do plugin SACRES 2.0 em execução. A *view* pode ser dividida em três partes: (i) a seção que seleciona as pastas e arquivos; (ii) a tabela com as *features* para alteração de configurações de *stakeholder*; e (iii) os botões de funcionalidade.

Grupos de *stakeholders* são representados por pastas dentro de um projeto do FeatureIDE e configurações de *stakeholders* são representadas por arquivos dentro dessas pastas. Na *view* primeiramente deve-se selecionar um grupo pertencente à lista de grupos associados ao FM. Depois é possível abrir configurações salvas dentro desse grupo ou criar uma nova configuração.

A tabela de configuração é dividida em três colunas, que listam o nome das *features*, o tipo da restrição associada a ela e o valor dessa restrição. O modelo de exemplo possui dez nodos na árvore, porém apenas oito estão sendo listados na tabela, pois o nodo raiz é abstrato não sendo uma *feature* propriamente dita e o nodo F1 representa uma *feature* obrigatória, assim sendo não é possível expressar preferências sobre ele. Ao estabelecer preferências, primeiramente o usuário seleciona o tipo (*hard* ou *soft*), e depois escolhe seu valor, caso seja *soft*.

No canto superior direito da *view* encontram-se quatro botões com as seguintes funções: apagar a configuração atual, salvar a configuração atual, atualizar as informações da *view* e executar os algoritmos de recomendação de configurações. Ao salvar uma configuração, o usuário deve escolher um nome para ela, digitando na caixa "Configuration". Quando executamos os algoritmos de recomendação, uma pasta com os resultados é criada e, caso não existam configurações válidas para serem recomendadas, uma mensagem de erro é exibida.

3.2. Estratégias de Escolha Social

Foram utilizadas sete estratégias baseadas na teoria da escolha social para a recomendação das configurações. Cada uma delas gera uma configuração de produto, de acordo com suas características. Sendo assim, após a execução do algoritmo da ferramenta, teremos até sete configurações como resultado, visto que estratégias diferentes podem gerar a mesma configuração. O algoritmo consiste de quatro passos principais: (i) tradução do FM e as preferências *hard* dos stakeholders para um CSP; (ii) geração de configurações

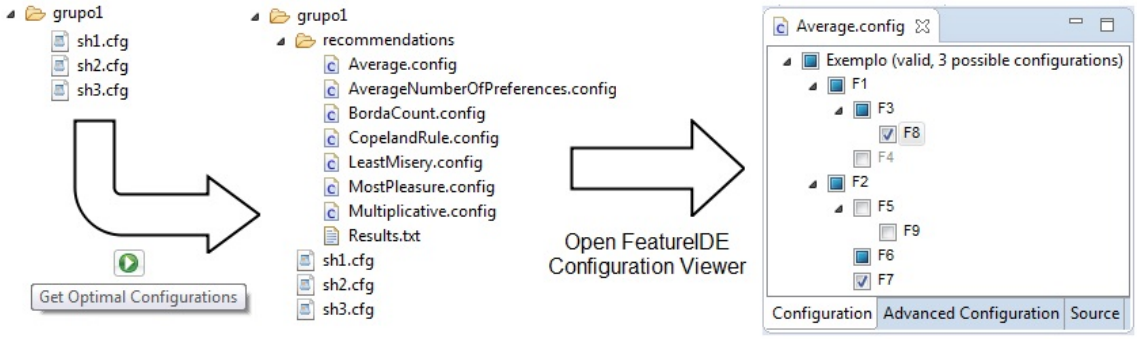

Figura 3. Arquivos e configurações salvas.

válidas utilizando o CHOCO; (iii) seleção da configuração ótima dentre as soluções viáveis de acordo com o critério de cada estratégia; e (iv) tradução das configurações ótimas para um arquivo de configuração seguindo o padrão do FeatureIDE.

As estratégias usadas são chamadas de: *Average (AVG), Average Number of Preferences (ANP), Borda Count (BC), Copeland Rule (CR), Least Misery (LM), Most Pleasure (MP) e Multiplicative (MUL)*. As estratégias *Average* e *Multiplicative* se preocupam em encontrar a satisfação global máxima dos *stakeholders*, a primeira calcula a média das satisfações individuais dos *stakeholders*, enquanto a segunda multiplica as satisfações individuais, e para ambas é selecionado o maior valor entre os calculados. A *Most Pleasure* se preocupa em encontrar a melhor satisfação de *stakeholder* possível, então ela seleciona o máximo entre o máximo de satisfações individuais de *stakeholders*. *Least Misery* se preocupa em não deixar os *stakeholders* muito insatisfeitos, o que pode ocorrer com as três estratégias acima. Para isso, ela seleciona o máximo entre o mínimo de satisfações individuais de *stakeholders*. A *Borda Count* usa a satisfação do *stakeholder* para ordenar as configurações. Assim temos uma classificação de configurações da satisfação mais alta (melhor) até a satisfação mais baixa (pior), e baseando-se nessa classificação cada configuração recebe um número de pontos. A pior configuração recebe 0 pontos, a segunda pior recebe 1 ponto e assim sucessivamente. Finalmente, os pontos de cada configuração individual de *stakeholder* são somados e o máximo é selecionado. *Copeland Rule* considera a relação de ordenação dada pela satisfação do *stakeholder*. Ela seleciona a configuração que tem a máxima diferença entre o quão frequente uma configuração ganha de outras configurações e o quão frequente ela perde, usando um sistema de votos. A *Average Number of Preferences* considera apenas a satisfação das restrições *soft*, ignorando os graus de preferência dos *stakeholders*. Ela seleciona a configuração com o número máximo de restrições *soft* satisfeitas.

A Figura 3 mostra uma parte do gerenciamento de arquivos feito pelo SACRES 2.0. Dentro de qualquer projeto do FeatureIDE existe uma pasta chamada “configs”. Nossa ferramenta trabalha com os arquivos contidos dentro dessa pasta. Cada subpasta presente nela representa um grupo de *stakeholders*, que podem ser criados, editados e apagados diretamente na *view* do projeto no Eclipse. Dentro das pastas de grupo encontram-se as configurações dos *stakeholders* associadas a ele. Estas configurações são baseadas no metamodelo descrito, são salvas com a extensão “.cfg” e podem ser exibidas usando a *view* da ferramenta (o conteúdo do arquivo “sh1.cfg” pode ser observado na Figura 2).

A. Configurações de stakeholders.

Feature	SH1	SH2	SH3
F2	+	0.6	+
F3			0.3
F4	-		
F5	-0.1		
F6			-0.15
F7	0.25	+	0.1
F8	0.5	0.2	
F9			

B. Recomendações das estratégias.

Total	AVG	ANP	BC	CR	LM	MP	MUL
Satisfação	2.2	2.2	2.1	2.2	2.1	2.05	2.2
Preferências	8	8	7	8	7	7	8

Tabela 1. Exemplo de recomendações usando preferências de stakeholders.

Na Figura 3 podemos também ver a pasta "grupo1" com três configurações diferentes. Após o clique no botão "*Get Optimal Configurations*" os algoritmos de recomendação são executados e uma nova pasta chamada "*recommendations*" é criada. Dentro dela estão os arquivos de configuração correspondentes a cada estratégia de escolha social no formato padrão do FeatureIDE ".config". Dessa forma é possível usar a funcionalidade de visualização de configurações do próprio FeatureIDE.

A Tabela 1(A) nos mostra as preferências de cada um dos três *stakeholders* pertencentes ao grupo. Os símbolos + e - representam restrições do tipo *hard* positivas e negativas, respectivamente. Os valores numéricos representam o grau de preferência de restrições do tipo *soft*. Na Tabela 1(B) exibimos alguns dos resultados para cada estratégia (coletados do arquivo "Results.txt" localizado na pasta "recommendations"). Ela mostra o total de satisfação dos *stakeholders*, ou seja, a soma das satisfações individuais e o número total de preferências satisfeitas para cada estratégia. Podemos observar que algumas estratégias conseguiram satisfazer todas as oito restrições do tipo *soft* presentes no grupo. Outras informações como o mínimo, o máximo, a média e o desvio padrão, tanto da satisfação individual de *stakeholder* quanto da quantidade de preferências para cada estratégia também podem ser encontradas no arquivo de resultados.

4. Trabalhos Relacionados

A ferramenta apresentada neste artigo permite a especificação de configurações associadas a uma LPS. Entretanto, essas configurações não são as usualmente utilizadas, mas configurações onde preferências *soft* e *hard* são especificadas. Além disso, recomendações são realizadas que resultam nessas configurações tradicionais. Assim, SACRES tem como parte de sua infraestrutura uma ferramenta para o gerenciamento de FMs. Foram analisadas diversas ferramentas a fim de encontrar o melhor candidato para a integração com o SACRES, entre elas citamos: SPLOT[3], XFeature[4] e FMP[5]. Estas ferramentas carecem de algumas funcionalidades desejadas no nosso plugin (como discutido na Seção 3), por isso foram descartadas e o FeatureIDE foi escolhido.

O plugin SPLConfig [Machado et al. 2014] também visa a recomendação automática de configurações ótimas maximizando a satisfação dos clientes, porém não suporta a participação de múltiplos stakeholders no processo de configuração de um mesmo produto. Ele mantém o foco em outros requisitos e limitações, como custo, benefício do consumidor e orçamento.

[3] `http://www.splot-research.org/`
[4] `http://www.pnp-software.com/XFeature/`
[5] `http://gsd.uwaterloo.ca/fmp`

5. Considerações Finais

Configuração de *feature model* é uma atividade importante para o desenvolvimento de produtos em uma LPS. O envolvimento de múltiplos *stakeholders* nesse processo pode torná-lo difícil devido aos possíveis conflitos entre suas preferências. Neste artigo apresentamos a ferramenta SACRES 2.0, um plugin para o Eclipse que visa dar suporte à recomendação de configurações considerando múltiplos *stakeholders*. A ferramenta utiliza estratégias de escolha social anteriormente propostas para buscar configurações que melhor satisfaçam o grupo de *stakeholders*.

O foco deste trabalho não foi a análise de cada estratégia usada, e sim as funcionalidades fornecidas pela ferramenta. Detalhes mais aprofundados sobre o desempenho das estratégias e avaliação do resultado das recomendações, com relação à satisfação individual e justiça, podem ser encontrados em [Stein et al. 2014]. Dois aspectos poderiam ser futuramente avaliados com relação à ferramenta: usabilidade e escalabilidade. Como um estudo anterior [Mendonca et al. 2009] concluiu que o processamento de *feature models* com um CSP Solver em geral não tem problemas de escalabilidade, temos indícios que o SACRES 2.0 não terá problemas de performance. Também, futuramente, possíveis melhorias podem ser adicionadas à ferramenta. Uma possibilidade seria considerar pesos para determinadas preferências dos *stakeholders* de acordo com sua expertise ou preferências sobre grupos de *features*, além de dar um maior suporte à colaboração distribuída dos *stakeholders*. Atualmente, as configurações dos *stakeholders* podem ser criadas em máquinas diferentes, entretanto os arquivos de configuração gerados devem ser agrupados para a geração de recomendações.

Reconhecimento

Este trabalho recebeu o apoio financeiro do programa PIBIC CNPq-UFRGS, com concessão de bolsa de iniciação científica ao primeiro autor deste trabalho.

Referências

Clements, P. and Northrop, L. (2002). *Software product lines: practices and patterns*, volume 59. Addison-Wesley Reading.

Czarnecki, K. and Eisenecker, U. (2000). *Generative Programming: Methods, Tools, and Applications*. Addison Wesley.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990). Feature-oriented domain analysis (foda) feasibility study. Technical report, DTIC Document.

Machado, L., Pereira, J., Garcia, L., and Figueiredo, E. (2014). Splconfig: Product configuration in software product line. In *CBSoft, Tools Session*, pages 1–8.

Mendonca, M. and Cowan, D. (2010). Decision-making coordination and efficient reasoning techniques for feature-based configuration. *Science of Computer Programming*, 75(5):311–332.

Mendonca, M., Wasowski, A., and Czarnecki, K. (2009). Sat-based analysis of feature models is easy. In *SPLC '09*, pages 231–240, USA. Carnegie Mellon University.

Stein, J., Nunes, I., and Cirilo, E. (2014). Preference-based feature model configuration with multiple stakeholders. In *SPLC '14*, pages 132–141, USA. ACM.