Adaptação de Processos de Software Utilizando *Situational Method Engineering* e *Search-base Software Engineering*

Humberto Streb[1]; Lisandra Manzoni Fontoura[1]

[1]Programa de Pós-Graduação em Informática – PPGI, Universidade Federal de Santa Maria – UFSM, Santa Maria, Brasil

{hstreb, lisandramf}@gmail.com

Abstract. *Tailoring process of software is a very complex task, involving many factors, including experience of process engineer, understanding the context of the project, among others. Such factors may cause not optimized and inconsistent results. This work proposes to optimize and add consistency to process tailoring using concepts of Search-base Software Engineering and Situational Method Engineering approach. The strategy used for recovery and validation of the fragments is the use of Octopus Model context, with fragments from the RUP, evolving proposal OSPTA.*

Resumo*. Adaptação ou elaboração de processos de software é uma tarefa de grande complexidade, envolvendo inúmeros fatores, como experiência do engenheiro de processos, entendimento do contexto do projeto, entre outros. Tais fatores podem causar resultados não otimizados e inconsistentes. Este trabalho propõe otimizar e adicionar consistência na adaptação de processos utilizando os conceitos de Search-base Software Engineering e a abordagem Situational Method Engineering. A estratégia utilizada para a recuperação e validação dos fragmentos é a utilização do contexto do Octopus Model, com fragmentos oriundos do RUP, evoluindo a proposta OSPTA.*

1. Introdução

Processos de software são definidos por Fuggeta (2000) como um conjunto de políticas, estruturas organizacionais, tecnologias, procedimentos, e artefatos que são necessários para conceber, implantar, e manter um produto de software. Processos de software tem grande impacto em um projeto, devido sua complexidade e necessidade de um grande conhecimento para sua elaboração. O reúso de processos, além de facilitar as tarefas de elaboração ou adaptação de novos processos, tende a aumentar a eficiência do processo ao aplicar práticas que foram bem-sucedidas em projetos anteriores. Os riscos de um projeto de software quando não gerenciados, podem aumentar o custo e extrapolar os prazos de um projeto, além de afetar a qualidade do software. Para solucionar esse problema é preciso elaborar um plano de gerência de riscos e adotar ações preventivas.

A atividade de adaptação de processos, seja para prevenir os riscos de um projeto ou não, pressupõe a consideração dos seguintes pontos: (i) os fragmentos selecionados precisam ser sequenciados em uma ordem coerente de execução; (ii) garantia que o processo gerado tenha consistência e inclua um conjunto de tarefas

definido como essencial a qualquer processo de software, segundo a literatura; (iii) a solução encontrada seja otimizada, sem repetição de atividades; (iv) o processo gerado possa ser reaproveitado futuramente; (v) tornar opcional a necessidade prévia de um processo padrão da organização; (vi) o tempo de resposta da solução proposta seja viável para uma aplicação real.

Neste trabalho, o processo de software é elaborado a partir de fragmentos de processos descritos segundo os conceitos propostos em *Situational Method Engineering* (SME) Henderson-Sellers e Ralyté (2010). O objetivo é gerar processos de desenvolvimento de software consistentes e otimizados utilizando os conceitos de *Search-base Software Engineering* (SBSE) Harmann e Jones (2001). A solução é encontrada, selecionando as melhores práticas com base nos riscos do projeto em questão. O processo gerado contém uma sequência lógica de execução. Os fragmentos de processo cadastrados na base de métodos são definidos segundo as atividades descritas pelo *Rational Unified Process* IBM (2006).

O artigo está estruturado da seguinte forma. Na Seção 2 é apresentado o referencial teórico sobre as técnicas utilizadas no trabalho. Na Seção 3 são descritos os trabalhos relacionados. Na Seção 4 é apresentada a abordagem proposta para adaptação de processos de software. Na Seção 5 são descritos os resultados obtidos e, por fim, na Seção 6 são apresentadas as considerações finais.

2. Referencial Bibliográfico

Adaptação de processos visa customizar processos de desenvolvimento de software para atender necessidades de uma organização ou projeto específicos. Cada projeto possui suas próprias características e necessita de técnicas e estratégias de desenvolvimento particulares Alegría (2011). As atividades de adaptação envolvem adicionar, excluir e/ou modificar elementos de processo, como por exemplo, papéis, atividades, artefatos, entre outros, Xu (2008).

A abordagem denominada *Situational Method Engineering* (SME), definida por Henderson-Sellers e Ralyté (2010), propõe a construção de um método/processo de desenvolvimento específico para cada projeto ou organização. Para tal tarefa, são definidos fragmentos de processo, descrevendo tarefas a serem executadas em um processo e artefatos elaborados durante a execução do processo, que são armazenados em uma base de métodos, que posteriormente são selecionados levando em consideração as características do projeto para compor novos processos.

O termo *Search-based Software Engineering* (SBSE) foi proposto por Harman e Jones (2001), e visa a aplicação de técnicas de busca para resolução de problemas na área de Engenharia de Software. Esta abordagem teve sua origem no fato de que muitas tarefas em Engenharia de Software podem ser formuladas como problemas baseados em busca Räihä (2010). O objetivo é encontrar, dentre uma ampla gama de possibilidades, uma solução que é suficientemente boa de acordo com uma função de qualidade apropriada. Tal técnica foi escolhida pelo potencial de aplicação na área de Engenharia de Software como exposto por Räihä (2010).

A técnica de busca utilizada neste trabalho é *Genetic Algorithms* (GA), proposta inicialmente por Holland (1975) e revisada principalmente por Goldberg (1989), que

consiste em uma analogia à evolução das espécies proposta por Darwin no século XIX, sendo destinada a solucionar problemas com um espaço de busca muito elevado. Na área de engenharia de software tal técnica foi utilizada com sucesso por Ngo-The e Ruhe (2009) para alocação de pessoas, visando otimizar o planejamento de *releases*.

A abordagem *Octopus SME Process Tailoring Approach* (OSPTA) proposta por Pereira (2012) armazena fragmentos de processos em um repositório com base nos conceitos de SME, e utiliza o *Octopus Model* Kruchten (2010) para representar o contexto do projeto e dos fragmentos. Nesta visão é utilizado o método *Analytic Hierarchy Process* (AHP) Saaty (1980) para priorizar a seleção dos fragmentos com base nos valores do *Octopus Model* e os riscos associados ao projeto, criando assim o processo adaptado.

O *Octopus Model* Kruchten (2010) destaca oito fatores para contextualizar o projeto, tais fatores influenciam o desenvolvimento de software, portanto devem ser considerados no momento de definir as atividades a serem incorporadas nos processos de software. Os fatores definidos no *Octopus Model* são: Tamanho (*Size*); Arquitetura Estável (*Stable Architecture*); Modelo de negócio (*Business Model*); Distribuição da equipe (*Team Distribution*); Taxa de mudança (*Rate of Change*); Idade do sistema (*Age of System*); Criticidade (*Criticity*); Controle (*Governance*). A Tabela 1 mostra os fatores que compõem o modelo e seus possíveis valores em projetos ágeis e planejados.

Tabela 1. Valores para definição de contextos com *Octopus Model* extraído de Pereira (2012)

Fator	**Valores Possíveis**		
	Características ágeis		Características planejadas
Tamanho	Pequeno	Médio	Grande
Arquitetura	Estável	Modificada	Móvel
Modelo de Negócios	In house ou sob medida para um cliente	Comercial	Componente de um grande sistema
Distribuição da Equipe	Mesmo local	Equipes diferentes	Distribuída geograficamente
Taxa de Mudança (% em um mês)	Mais que 30	Entre 10 e 30	Menos que 10
Idade do Sistema	Novo desenvolvimento	Manutenção	Evolução de sistema legado
Criticidade	Perda de conforto	Perda de dinheiro	Mortes
Controle	Dinâmico / flexível	Regras simples	Mecânico / formal

A técnica *Analytic Hierarchy Process* (AHP) Saaty (1980) é um modelo matemático para apoio à teoria de decisão, e é utilizada em ambientes complexos, onde muitas variáveis precisam ser consideradas para seleção e priorização de alternativas.

3. Trabalhos Relacionados

Nesta sessão são apresentados os trabalhos relacionados e as vantagens e desvantagens do trabalho proposto.

Bryers et al. (2009) faz uso das técnicas de SBSE para montar um método de adequação das atividades de segurança chamado S^3P (*Sustainable Software Security Process*) e utiliza *Analytic Hierarchy Process* (AHP) para priorizar as atividades. Para selecionar o melhor conjunto de atividades que atendam objetivos de segurança são empregadas as meta-heurísticas Busca Dispersa (*Scatter Search*) combinada com Busca Tabu, baseados em modelos de vulnerabilidades.

Magdaleno (2010) utiliza *Context-Based Process Line Engineering* (CBPLE) para a resolução do problema. O problema foi modelado através do *Balance in Process Tailoring* (BPT) e os componentes do processo são recuperados a partir de características do contexto, que foi dividido entre *Collaboration* (com características ágeis) e *Discipline* (com características planejadas), no passo seguinte os componentes são combinados, podendo sofrer alterações do gerente de processo, e por fim são sequenciados através das regras de leitura e escrita dos artefatos de cada componente.

O trabalho de Guo et al. (2012) apresenta uma abordagem baseada em *Software Product Line* (SPL), com foco na consistência do processo gerado, utilizando ontologias para formalizar as *Features Models* (FM)[1], definindo as restrições às FM, e um conjunto de operações primitivas que modificam as FM e são responsáveis pela evolução, analisando seu impacto na consistência da FM. A consistência é verificada a partir da última mudança em uma FM em vez de verificar a consistência global da FM resultante.

A Tabela 4 apresenta um quadro comparativo entre os trabalhos relacionados. O conceito apresentado neste artigo gera um fluxo de atividades, não necessita de um processo padrão da organização e utiliza técnicas de otimização.

Tabela 2. Quadro comparativo dos trabalhos relacionados.

Contribuição	Bryers et al. (2009)	Magdaleno (2010)	Guo et al. (2012)
Gera um fluxo de atividades	Não	Sim	Sim
Necessidade prévia de processo padrão da organização	Sim	Sim	Sim
Utiliza técnicas de otimização	Sim	Sim	Não

Entre as vantagens deste trabalho estão a validação do processo gerado; e a eliminação da necessidade de um processo padrão organizacional. Como desvantagem, está a não utilização de Métodos Ágeis na criação da base de métodos. A adição de Métodos Ágeis pode ser expandida futuramente, necessitando apenas da inserção de fragmentos, elaborados a partir de métodos e práticas ágeis, na base de métodos. Ao passo que, a proposta foi concebida de forma a ser genérica o suficiente para contemplar fragmentos de diferentes metodologias.

4. Evolução da Abordagem OSPTA

Este trabalho visa estender a abordagem OSPTA, citada anteriormente, adicionando busca heurística para recuperar os fragmentos, a fim de aumentar a chance de encontrar uma solução otimizada, além de implementar a validação no momento da busca, recuperando apenas fragmentos válidos conforme o contexto e considerando os fragmentos já selecionados. A verificação da consistência dos fragmentos é realizada considerando a obrigatoriedade ou não de cada artefato de entrada e saída de um fragmento.

1 *Feature Models* representam as relações entre os pontos comuns e variáveis em um *Software Product Line*.

Outro objetivo é adicionar uma sequência de execução a solução. Em OSPTA, são selecionados os fragmentos que atendem aos riscos definidos para o projeto e que são adequados ao contexto definido, e o fluxo de execução desses fragmentos (modelagem dinâmica) não é considerado. Para possibilitar a definição do fluxo de execução, o metamodelo do OSPTA foi estendido para representar as ligações entre os fragmentos e os artefatos para que seja possível identificar qual a precedência entre os fragmentos.

As atividades definidas na abordagem OSPTA foram reutilizadas neste trabalho. A recuperação e priorização dos fragmentos que originalmente eram duas atividades, foram unidas em uma única atividade, denominada *Recuperar fragmentos de método*. A atividade *Construir processo específico de software para o projeto* foi removida. Na Figura 1 é exibida a sequência das atividades propostas por este trabalho.

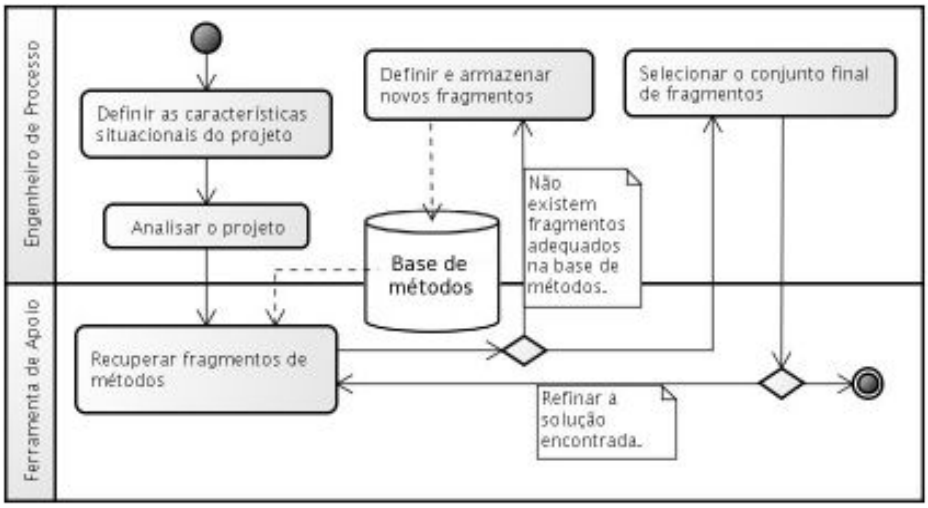

Figura 1. Sequência das atividades propostas neste trabalho

Na atividade *Definir características situacionais do projeto,* o engenheiro de processo contextualiza a situação do projeto, por definir valores adequados às características do projeto aos fatores propostos pelo *Octopus Model* (Tabela 1).

Na atividade *Analisar o projeto,* o engenheiro de processo define o(s) critério(s) estabelecido(s) para a adaptação. Neste trabalho limitou-se a utilizar como critério de adaptação a "Prevenção a Riscos do Projeto". Portanto, ao associar um risco ao projeto, o engenheiro de processos informa o potencial de perda do risco (*loss*) e a probabilidade de ocorrência (*probability*), que multiplicados geram o fator de exposição ao risco (*riskExpouse*) associados ao projeto. São selecionados todos os riscos com um percentual maior que determinado valor informado pelo gerente de projeto.

Com base nas informações sobre os riscos, na atividade *Recuperar fragmentos de métodos* são recuperados fragmentos da base de métodos que visam prevenir os riscos priorizados para o projeto. Nessa atividade concentra-se o esforço deste trabalho, no qual o método de busca dos fragmentos foi alterado, adicionando busca heurística para recuperar os fragmentos com base no contexto situacional e no(s) critério(s) de seleção, retornando apenas fragmentos válidos, respeitando a sequência de execução destes, conforme as regras definidas na leitura e escrita de seus artefatos.

Por meio da atividade *Definir e armazenar novos fragmentos,* novos fragmentos

podem ser inseridos na base de métodos e fragmentos existentes podem ser alterados.

A atividade *Selecionar o conjunto final de fragmentos* possibilita a seleção dos fragmentos usando o algoritmo proposto. O engenheiro de processos pode aprovar ou alterar o conjunto final de fragmentos de métodos, podendo executar novamente o algoritmo de busca para refinar a solução.

4.1. Modelagem do Problema

O problema é definido como a seleção do conjunto de fragmentos que melhor se adapta aos fatores do *Octopus Model* e aos critérios de adaptação do projeto. Esse problema foi modelado como encontrar o melhor caminho em um grafo direcionado, dado um grafo $G = (V, E)$, onde:

– V é o conjunto de vértices (*vertices*), que representam os fragmentos de método, que são chamados apenas de fragmento no restante do trabalho;

– E é o conjunto das arestas (*edges*), que representam os artefatos de entrada e saída de cada fragmento.

A Figura 2 mostra duas formas de representação de um mesmo grafo, em (a) é apresentado como uma matriz de adjacência, e em (b) a forma gráfica de um grafo, onde os vértices representam os fragmentos de método e os artefatos de entrada obrigatórios são representados pelas arestas em forma de linha contínua e os não obrigatórios em linha tracejada.

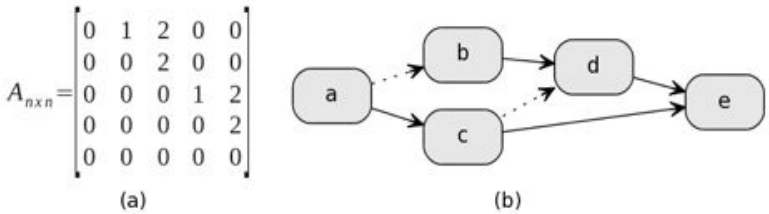

Figura 2. Representações do grafo proposto para o problema

A descrição formal da matriz de adjacência $A_{n\,x\,n}$ é apresentada na Fórmula (1):

$$a_{ij} = \begin{cases} 2, (i,j) \in ME \\ 1, (i,j) \in NME \\ 0, (i,j) \notin E \end{cases} \qquad (1)$$

Seja ME um conjunto de todas as arestas obrigatórias (*Mandatory Edge*), e cada aresta $ME_{ij} \in E$.

Seja NME um conjunto de todas as arestas não obrigatórias (*Non-Mandatory Edge*), e cada aresta $NME_{ij} \in E$. Minimizar:

$$\sum_{j=1}^{n} c_j x_j \qquad (2)$$

Sujeito a:

$$\sum_{j=1}^{n} a_{ij} x_j \qquad (3)$$

O cálculo do custo do fragmento *j* é obtido através da expressão matemática exposta na Fórmula (4), onde a soma de TCA_j com VC_j tende a 2, e a proposta do trabalho é encontrar o caminho com o menor custo, a fórmula é descrita com o número 2 menos a soma de TCA_j com VC_j.

$$c_j = 2 - (TCA_j + VC_j) \qquad (4)$$

TCA_j – Total dos Critérios de Adaptação do fragmento *j*, é obtido pelo somatório da divisão do Valor do Critério de Adaptação (VCA) de todos os critérios de adaptação selecionados pelo engenheiro de processos associados ao fragmento *j*, por cem, dividido pelo número de critérios de adaptação associados ao fragmento *j* (*k*), exposto na Fórmula (5).

$$TCA_j = \frac{1}{k} \sum_{i=0}^{n} max(0, \frac{VCA_i}{100}) \qquad (5)$$

O VCA no caso dos riscos do projeto, é o valor da Exposição ao Risco Coppendale (1995), que é obtido através do produto do potencial de perda com a probabilidade de ocorrência associados ao risco, tais valores são definidos previamente pelo gerente de projetos. A probabilidade varia de 0 (probabilidade baixa) a 10 (probabilidade alta), já o potencial de perda varia de 0 (nenhum aumento no custo ou tempo do projeto) a 10 (representa um aumento significativo no tempo ou no custo).

VC_j – Valor do Contexto do fragmento *j,* é obtido com base nos valores de contexto do *Octopus Model* extraídos utilizando a técnica AHP, conforme a definição da Abordagem OSPTA.

4.2. Definição do Algoritmo Genético

Na codificação do Algoritmo Genético (*Genetic Algorithm*) proposto nesse trabalho, cada indivíduo possui um cromossomo, que é a representação da lista dos fragmentos selecionados, onde cada fragmento é mapeado em um gene, e estes estão ordenados seguindo a ordem definida pelos artefatos de entrada e saída de cada fragmento. A codificação de um indivíduo é exposta na Figura 3.

Figura 3. Representação de um Indivíduo no Algoritmo Genético proposto

O funcionamento do Algoritmo Genético consiste em quatro operações básicas: cálculo de aptidão (*fitness evaluation*), seleção (*selection*), cruzamento (*crossover*) e mutação (*mutation*), a população inicial geralmente é gerada aleatoriamente.

Existem alguns parâmetros do Algoritmo Genético que afetam drasticamente seu desempenho. Neste trabalho são utilizados os seguintes parâmetros: *taxa de cruzamento* é a porcentagem da população que será selecionada para o cruzamento, *taxa de mutação*

é a probabilidade de um gene sofrer mutação, *tamanho da população* é o número de indivíduos por geração, o *número máximo de gerações* é o critério de parada do Algoritmo Genético e o *número de pontos de corte* que indica em quantas partes o cromossomo será fatiado.

Na Figura 4 é mostrado o fluxograma básico do funcionamento do Algoritmo Genético, essas atividades são detalhadas a seguir:

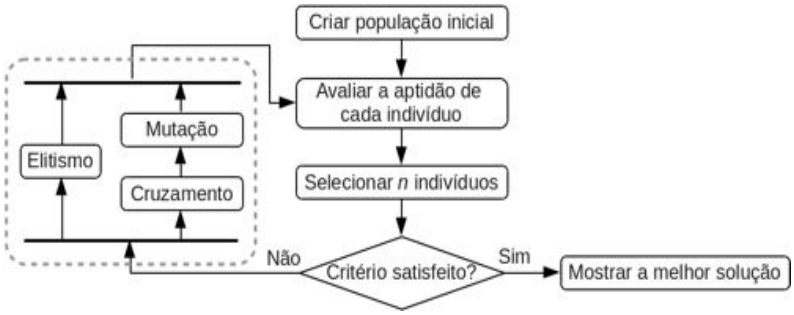

Figura 4. Atividades realizadas no Algoritmo Genético proposto

Para *Criar a população inicial* são seguidos os seguintes passos: (i) é realizada uma consulta na base de métodos buscando todos os fragmentos; (ii) um número aleatório desses fragmentos são selecionados, e cada fragmento desta seleção é associado a um gene de um novo indivíduo. O passo (ii) é repetido até que o tamanho da população seja atingido.

Avaliar a aptidão de cada indivíduo consiste em percorrer cada gene do indivíduo calculando seu custo conforme a Fórmula (4), e somando este resultado ao custo total do indivíduo. A fim de aumentar a probabilidade de selecionar os melhores indivíduos, o cálculo da aptidão (*fitness*) o custo do indivíduo é elevando ao quadrado, visando criar uma diferença exponencial entre os indivíduos. Para melhorar o desempenho do AG, no momento da avaliação, ao encontrar um fragmento que possui um artefato de saída obrigatório, verifica-se se todos os fragmentos que possuem o mesmo artefato de entrada obrigatório estão presentes no cromossomo, caso não estejam todos, os que faltam são adicionados na posição subsequente. Na Figura 5 é exibido como ocorre a adição de fragmentos pelas regras contidas nos artefatos.

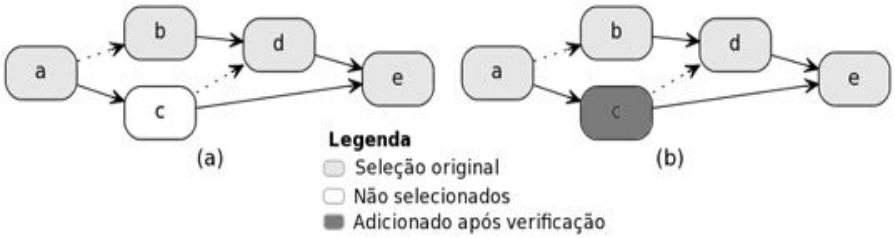

Figura 5: Adição de fragmento com artefato obrigatório no momento de avaliar a aptidão

Na etapa *Selecionar n indivíduos* é realizada a seleção dos indivíduos utilizando a técnica de seleção por roleta (*Roulette Wheel*[2]), que apresentou melhores resultados nos testes computacionais.

2 *Roulette Wheel* consiste em atribuir ao indivíduo a chance de ser selecionado proporcional ao seu *fitness*, em relação à somatória da aptidão de toda a população, onde, quanto maior a adaptação do indivíduo, maior a chance de ser selecionado.

Após selecionar os melhores indivíduos, é verificado se o critério de parada do algoritmo foi satisfeito. Isto é, o número máximo de gerações definido previamente foi atingido. Se sim, o algoritmo termina e mostra a melhor solução, se não parte para o cruzamento dos indivíduos selecionados, executando as etapas *Cruzamento, Mutação* e *Elitismo* novamente.

A etapa *Cruzamento* é responsável pela recombinação de características dos pais, gerando dois novos indivíduos. São escolhidos dois indivíduos da geração atual, determina-se os pontos de corte, troca-se a carga genética entre os dois indivíduos gerando os filhos. Na Figura 6 é mostrado um exemplo de cruzamento dois pontos.

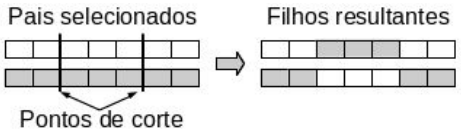

Figura 6: Cruzamento n pontos, com n=2

A *Mutação* é realizada no momento da troca de material genético dos pais, ou seja, a cada gene que será transferido de um dos pais para o descendente, é gerado um número aleatório de zero até cem, se este for menor que a taxa de mutação escolhida o gene é alterado, trocando o fragmento contigo neste por outro.

O *Elitismo* é a cópia dos dois melhores indivíduos para a geração seguinte, garantindo que o algoritmo não perca a melhor solução já encontrada.

Após chegar a geração esperada o algoritmo encerra e mostra o resultado, na etapa *Mostrar a melhor solução,* gerando um *script* na linguagem DOT Graphviz (2012) contendo o esquema dos fragmentos selecionados e os artefatos que os ligam, formando um grafo. O *script* gerado é executado pela ferramenta dot gerando uma imagem do grafo.

5. Estudo de Caso

Uma empresa de desenvolvimento de software deseja adaptar o processo de software para um novo projeto. O gerente de projeto destacou como riscos para o projeto: “não entendimento dos requisitos” e “instabilidade dos requisitos”.

O primeiro passo é contextualizar a situação do projeto, o engenheiro de processos atribui aos fatores propostos pelo *Octopus Model* valores apropriados as características do projeto, e o peso relativo entre os fatores. A Tabela 3 apresenta os valores atribuídos, bem como os valores dos pesos relativos entre os critérios.

Tabela 3. Contexto do *Octopus Model* do projeto

Critério	Valor	Peso Relativo
Tamanho	Grande	0,19
Criticidade	Perda de dinheiro	0,19
Arquitetura estável	Estável	0,19
Modelo de negócio	Comercial	0,19
Distribuição da equipe	Mesmo local	0,07
Taxa de mudanças (% em um mês)	Menos que 10	0,07
Idade do sistema	Novo desenvolvimento	0,07
Controle	Mecânico/formal	0,3

O próximo passo é definir o critério de adaptação, o engenheiro de processos associa um risco ao projeto informando o potencial de perda do risco e a probabilidade de ocorrência, gerando a exposição ao risco, os valores definidos para o projeto são expostos na Tabela 4.

Tabela 4. Contexto dos riscos do projeto

Risco	Potencial de Perda (a)	Probabilidade de Ocorrência (b)	Exposição ao Risco (a*b)
Não entendimento dos requisitos	7	10	70
Instabilidade de requisitos	3	1	3

Para exemplificar o cálculo do Total dos Critérios de Adaptação (TCA), considere que o fragmento "Entregas frequentes e regulares" está associado ao risco "Não entendimento dos requisitos", que possui valor de Exposição ao Risco é igual a 70 (Tabela 4). O seu VCA (Valor do Critério de Adaptação) é igual a 70. Aplicando-se a Fórmula (5), divide-se o VCA por cem, como o número de critérios de adaptação igual a um, o resultado é 0,7.

Na fase de recuperação dos fragmentos de métodos, o Algoritmo Genético busca os fragmentos que melhor se adaptam ao contexto do projeto. O resultado obtido é apresentado na Figura 7 na forma de um grafo. O grafo apresenta ligações obrigatórias (linha contínua) e não obrigatórias (linha tracejada). Essa informação é utilizada apenas no momento de execução do algoritmo, pois para a solução encontrada não faz diferença se um artefato é ou não obrigatório, se ele foi selecionado pelo algoritmo, será executado no processo.

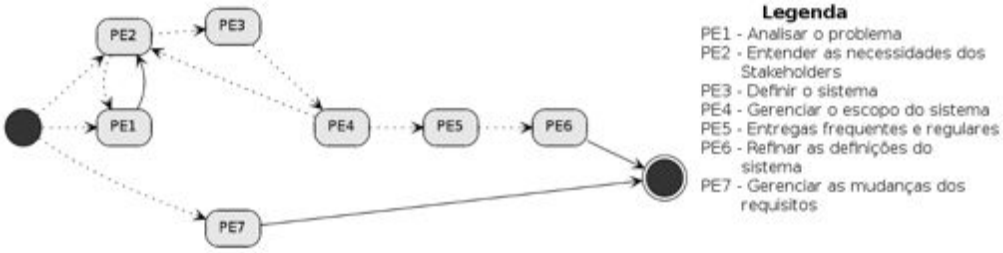

Figura 7: Processo gerado pelo Algoritmo Genético

O fragmento "Entregas frequentes e regulares" está associado ao risco "Não entendimento dos requisitos" e o fragmento "Gerenciar as mudanças dos requisitos" está associado ao risco "Requisitos instáveis". Na solução encontrada a ligação obrigatória no grafo entre os fragmentos "Analisar o problema" e "Entender as necessidades dos *Stakeholders*" é porque o artefato "Visão" é um artefato obrigatório de saída no primeiro e um artefato obrigatório de entrada no segundo. Já as outras ligações obrigatórias são as ligações finais, que servem apenas como referência para o término do processo.

Foram realizados testes computacionais para calibrar os parâmetros do Algoritmo Genético. O número de gerações e o tamanho da população, provocam uma grande influência tanto no desempenho, quanto no resultado do algoritmo, portanto para a realização dos testes estes se mantiveram constantes, o tamanho da população foi

fixado em 4.000 e como critérios de parada 1.000 gerações. Com estes valores a média do tempo de execução do algoritmo foi de 105 segundos, se aumentar esses valores se chega a uma melhor solução, mas o tempo de execução aumenta quase vertiginosamente, valores menores a solução é inferior, mas o tempo de execução é muito menor, por exemplo com população de 1.000 indivíduos e 1.000 gerações executa em menos de 15 segundos.

A taxa de mutação que apresentou os melhores resultados foi 0,001. A taxa de cruzamento que resultou a médio e longo prazo as melhores soluções foi de 0,85. Quanto maior esse valor, mais rapidamente é encontrada uma boa solução, mas ao longo prazo a população evoluía de forma lenta, e ao final da execução a resposta encontrada era inferior, quando comparada com a encontrada pela taxa escolhida. Taxas menores que 0,85 evoluíam mais lentamente e não surtiam o efeito esperado a longo prazo. Por fim, o número de pontos de corte que gerou os melhores resultados foi 4.

6. Conclusão e Trabalhos Futuros

Neste trabalho foi apresentada uma abordagem para adaptação de processos de software baseado nos conceitos de *Situational Method Engineering* e *Search-based Software Engineering,* utilizando o contexto do *Octopus Model* e *Analytic Hierarchy Process* para a priorização dos fragmentos com base nos riscos do projeto.

Na abordagem proposta é utilizada uma visão multicritério para selecionar e priorizar os fragmentos, sendo facilmente estendida para outros critérios de seleção, genérica o suficiente para abranger também o contexto do projeto, aumentado a chance de uma solução otimizada.

Entre as contribuições pode-se destacar a validação do processo gerado, através da adição de obrigatoriedade aos artefatos de cada fragmento; o sequenciamento da solução gerada, ainda através da leitura e escrita dos artefatos; a utilização de técnicas de *Search-based Software Engineering* aliadas a *Situational Method Engineering* e a refatoração e expansão da base de dados de fragmentos criada por Pereira (2012).

Como trabalhos futuros se pode citar: adicionar novos critérios de adaptação, para contemplar as demais variáveis envolvidas no planejamento e desenvolvimento de um projeto; a aplicação de outras meta-heurísticas, além de Algoritmos Genéticos; a criação de uma ferramenta de auxílio na tomada de decisão, com informações visuais da solução.

Referências

Alegría, J., Bastarrica, M., Quispe, A. and Ochoa, S. (2011) An MDE approach to software process tailoring. In: Proceedings of the 2011 International Conference on Software and Systems Process, p. 43-52.

Bryers, D. And Shahmehri, N. (2009) Prioritization and Selection of Software Security Activities. In: Proceedings of International Conference on Availability, Reliability and Security (ARES09).

Coppendale, J. (1995) Managing Risk in Product and Process Development and Avoid Unpleasant Surprises. Engineering Management Journal, New York, v.5, n.1, p. 35-

38.

Fuggeta, A. (2000) Software Process: A Roadmap. In: International Conference On Software Engineering, 22., 2000, New York. Proceedings... New York: ACM Press.

Graphviz. (2012) Documentation. Disponível em: <http://graphviz.org/Documentation.php/>. Dezembro de 2012.

Goldberg, D. E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Reading MA: EUA, Addison Wesley.

Guo, J., Wang, Y., Trindad, P. And Benavides, D. (2012) Consistency maintenance for evolving feature models. Expert Systems with Applications, Volume 39, Issue 5, April, Pages 4987-4998, ISSN 0957-4174, 10.1016/j.eswa.2011.10.014.

Harman, M. And Jones B. F. (2001) Search Based Software Engineering. In: Information & Software Technology, vol. 43, no. 14, pp. 833-839.

Henderson-Sellers, B. And Ralyté, J. (2010) Situational Method Engineering: State-of-the-Art Review. Journal of Universal Computer Science. vol. 16, no. 3, pp. 424- 478.

Holland, J. H. (1975) Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. University of Michigan Press.

IBM. (2006) Rational Unified Process (software). version 7.0. USA: IBM Rational.

Kruchten, P. (2010) Contextualizing Agile Software Development. In: Proceedings of the EuroSPI Conference, p. 1-12.

Magdaleno, A.M. (2010) Optimization-based Approach to Software Development Process Tailoring. In: Second International Symposium on Search Based Software Engineering (SSBSE), p. 40-43.

Ngo-The, A. And Ruhe, G. (2009) Optimized Resource Allocation for Software Release Planning. Software Engineering, IEEE Transactions on, vol.35, no.1, pp.109-123.

Pereira, Guilherme V. (2012) Abordagem Multicritérios para Adaptação de Processos de Software Baseada em Situational Method Engineering. Dissertação de Mestrado, Universidade Federal de Santa Maria.

Pereira, G. And Fontoura, L. (2012b) Defining Agile and Planned Method Fragments for Situational Method Engineering. Em: VIII Simpósio Brasileiro de Sistemas de Informação. Fortaleza, Brasil.

Räihä , O. (2010) A survey on search-based software design. Computer Science Review, vol. 4, Issue 4, pp 203-249.

Xu, P. And Ramesh, B. (2008) Using Process Tailoring to Manage Software Development Challenges. IT Professional, vol.10, no.4, pp.39-45.