

Uma Ferramenta para Auxiliar na Execução de Processos de Software Modelados em SPEM

Douglas Montanha Giordano¹, Eduardo Bruning²

¹ Mestrado de Ciência da Computação – Universidade Federal de Santa Maria(UFSM)
Av. Roraima, 1000 – CEP 97105-900 – Santa Maria – RS

douglasmontanhagiordano@gmail.com, eduardo.bruning@gmail.com

Resumo. *Comumente vários processos são definidos para guiar o desenvolvimento de um software. Atualmente existem várias ferramentas que oferecem suporte a determinadas etapas do processo de software. A partir de uma pesquisa podemos visualizar a necessidade de uma ferramenta que apoie a etapa de execução do processo. O objetivo deste trabalho é oferecer um software de apoio a execução de processos modelados em SPEM. Para isso construímos uma ferramenta que gera diagramas dinâmicos representando o processo onde a entrada é o documento no formato SPEM. Esta ferramenta possibilita o acompanhamento da execução de cada etapa do processo e também o arquivamento dos artefatos gerados em cada etapa.*

1. Introdução

A demanda de software atualmente cresce cada vez mais. Em todo lugar podemos perceber softwares de todos os tipos em diferentes dispositivos, gerando uma grande demanda de produtos. Com a produção de software cada vez mais agressiva, a necessidade de técnicas que auxiliem sua produção em menos tempo e com maior qualidade é requerida [Larman 2007].

O processo de software está bastante ligado com a qualidade e tempo de construção do produto, podendo causar o fracasso do projeto ou guiá-lo ao sucesso [Sommerville 2007]. Antes do desenvolvimento de um software, é comum definir como será o processo de sua construção. A definição é apenas uma das etapas relacionadas ao processo de software, sendo ligada a outras características como a sua melhoria e execução.

Com objetivo de oferecer uma ferramenta de auxílio a etapa de execução de processos, esse artigo apresenta a construção de uma ferramenta. Essa ferramenta auxilia nas principais etapas da execução do processo, como o controle de fluxo das atividades, e o controle dos artefatos de softwares gerados em cada uma delas.

Na seção 2 é apresentado os principais conceitos envolvidos na modelagem de processos. Na seção 3 é apresentado um mapeamento das ferramentas que auxiliam algumas etapas do processo de software. Já na seção 4 é descrito alguma das etapas do desenvolvimento da ferramenta e as principais estruturas envolvidas na implementação. Por ultimo na seção 5 é mostrado a ferramenta, e um pouco de suas funcionalidades.

2. Fundamentação Teórica e Tecnológica

2.1. Modelagem de processos

A modelagem de um processo de software é realizada para possibilitar o entendimento e a padronização do processo aplicado durante o desenvolvimento de um software. Podemos destacar alguns objetivos para modelar processos de softwares [GENVIGIR et al. 2003], como:

- Possibilitar a comunicação e o entendimento efetivo do processo;
- Facilitar a reutilização do processo (padronização);
- Apoiar a evolução do processo;
- Facilitar o gerenciamento do processo.

Vários fatores levam para padronização dos processos, alguns dos principais são: permitir treinamento, gerenciamento, revisões, ferramentas de suporte, utilizar processos padronizados, contribuir para a melhoria dos processos na organização com as experiências de projeto e fornecer uma base estrutural para medição do processo [GENVIGIR et al. 2003].

A classificação dos elementos dentro de um processo de software pode ser classificado como elementos e relacionamentos. Um modelo de processo concreto consiste de instâncias desses tipos de relacionamentos e elementos. O processo consiste em atividades de cooperação concorrentes abrangendo atividades de manutenção, desenvolvimento, segurança, qualidade e gerenciamento de projeto. Os elementos e os relacionamentos permitem uma modelagem rústica de um processo de software. Sendo assim, uma linguagem de modelagem de processo de software deve fornecer recursos de linguagem para modelar esses elementos básicos e avançados bem como seus inter-relacionamentos. As variantes podem ser classificadas de acordo com características. Como não é possível abranger todas as características dentro de uma linguagem de especificação, deve-se tomar a decisão do que está relacionado a fase do meta-processo e quem está lidando com o modelo do processo na construção do modelo da linguagem [GENVIGIR et al. 2003].

2.2. Software Process Engineering Metamodel – SPEM

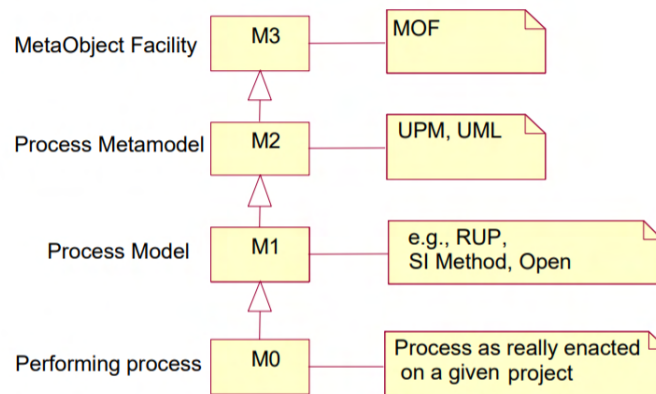
O SPEM é o metamodelo proposto pela da OMG¹ para a descrição de um processo concreto de desenvolvimento de software ou uma família de processos de desenvolvimento de software. O SPEM surgiu com o objetivo de unificar diferentes metodologias propostas para a modelagem de processos. Ele utiliza a orientação a objeto para modelar uma família relacionada de processos de software e usa a UML como notação.

Na Figura 1 é exibido as quatro arquiteturas de modelagem definidas pela OMG. O SPEM se encontra no nível M2 da arquitetura, é definido usando um subconjunto da UML, semelhante ao que a UML realiza com o MOF, parte dela utiliza as facilidades implementadas e apoiadas pelo MOF [GENVIGIR et al. 2003].

Segundo a proposta do SPEM quando um processo é executado, aplicado no mundo real ele está no nível de M0. Um processo definido está no nível M1, como por exemplo o Processo Unificado Rational – RUP. O SPEM pode ser usado para definir todos os tipos de processos, incluindo os focados no uso específico da UML. A UML é

¹Google Scholar <http://www.omg.org/>

Figura 1. Gráfico do ano das publicações



definida por um metamodelo, que é definido como uma instância do metamodelo MOF, localizado em M3. Como foi relatado, apenas um subconjunto da notação gráfica da UML é usado para descrever este metamodelo. O *metamodel* SPEM foi elaborado similarmente, é formalmente definido como uma extensão de um subconjunto da UML chamado *SPEM_Foundation* [GENVIGIR et al. 2003].

3. Trabalhos Relacionados

Para investigar os trabalhos relacionados a esta pesquisa foi utilizado um mapeamento sistemático da literatura. Ele consiste em três etapas, a definição do protocolo do mapeamento, a execução do protocolo e a análise dos resultados.

3.1. Protocolo do mapeamento

O objetivo do protocolo é estabelecer o estado da arte das ferramentas que dão apoio a execução de processos baseados em SPEM. Baseado nisso, foi definido apenas uma questão de pesquisa para investigar nos trabalhos selecionados, sendo ela:

- Qual a função e objetivo das ferramentas?

O mecanismo de busca escolhido foi o Google Scholar ², levando em consideração que ele indexa a maior parte das bases de dados. A *string* de busca utilizada foi “software process”+“tool” OR “tools” OR “CASE” + “software & systems process engineering metamodel” OR “SPEM”. Para filtrar a busca, definimos os seguintes critérios de inclusão.

- Artigos completos publicados de 2012 até 2017 e com no mínimo 6 (seis) páginas;
- Artigos que abordam sobre ferramentas de suporte ao SPEM.

Para retirar trabalhos que não são interessantes para esta pesquisa definimos dois critérios de exclusão.

- Artigos que sejam outros mapeamentos ou revisões sistemáticas;
- Artigos repetidos ou versões resumidas de trabalhos completos.

De acordo com os trabalhos retornados, o modo de classificação definido foi pelo ano e o veículo de publicação.

²Google Scholar <http://scholar.google.com>

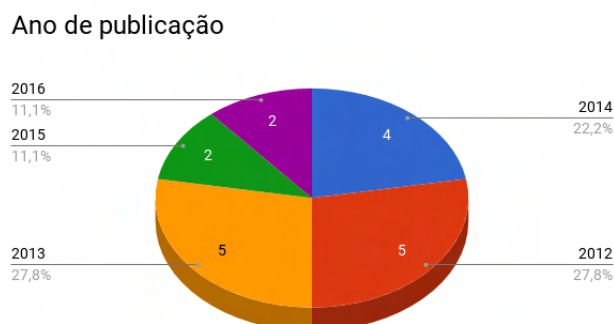
3.2. Resultado

A execução do protocolo retornou 2060 trabalhos, com a aplicação dos critérios de inclusão este número reduziu para 726 e após aplicar os critérios de exclusão 19 trabalhos foram selecionados para leitura e análise mais aprofundada. Dentre os trabalhos, nenhum apresentou ferramentas para apoiar a execução de processos, servindo de suporte para guiar e gerenciar as etapas do processo em sua execução.

3.2.1. Classificação dos resultados

Para visualizar a quantidade de publicações pelos anos, é exibido na Figura 2 os trabalhos selecionados de acordo com seu ano de publicação. A grande maioria está entre os anos de 2012 a 2014, sendo a concentração maior no ano de 2012, seguido de 2013 e 2014. Os anos de 2015 e 2016 são equivalentes em relação a quantidade de publicações. Desta forma, nos últimos anos as publicações sobre este assunto apresentam uma queda de quantidade. Isso é visível apenas observando a Figura 2, sendo que não existem trabalhos selecionados do ano de 2017.

Figura 2. Gráfico do ano das publicações



Com a intenção de identificar onde ocorrem a maior parte das publicações dos trabalhos selecionados, eles foram classificados de acordo com seu veículo de publicação, exibidos na Figura 3. A maior quantidade está na fatia chamada outros, que engloba trabalhos de bases de universidades e de veículos de menor expressão. Esse agrupamento foi realizado para melhorar a exibição dos dados. Dentre os veículos mais conhecidos a IEEE lidera com 4 trabalhos (ou 21,1%) seguida pela Elsevier com 3 trabalhos (ou 15,8%).

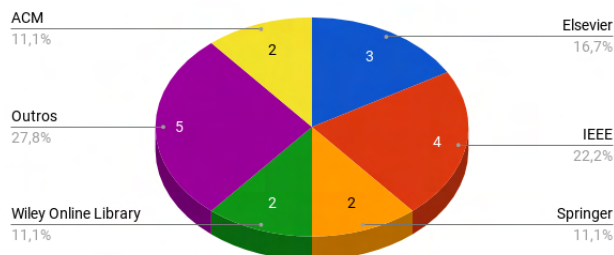
Após uma exibição geral dos trabalhos, a seguir são exibidas as respostas dos trabalhos para a questão de pesquisa definida no protocolo.

3.2.2. Qual a função e objetivo das ferramentas?

Kuhrmann et al. (2043) apresentam a ferramenta Process Enactment Tool Framework (PET), que tem como objetivo transformar determinados modelos de processo formal em

Figura 3. Gráfico do veículos das publicações

Veículos de publicação



um formato que as ferramentas de projetos de processos possam interpretar, ou seja, O PET é um instrumento para importar processos com base em um metamodelo e fornecer exportações para um ambiente de projeto específico [Kuhrmann et al. 2014]. Ellner et al. (2012) apresentam uma cadeia de ferramentas integrada baseada em eSPeM e Eclipse. Elas suportam modelos de processo baseados em eSPeM e também orienta e apoia a equipe de projeto a trabalhar de acordo com o processo de maneira sensível ao contexto. Elas automatizam o trabalho repetitivo e pesado como verificação de conformidade do processo ou acompanhamento de progresso [Ellner et al. 2012]. Já Hurtado et al. (2013) propõem uma abordagem orientada por modelo para especificação e configuração de linhas de processo de software [Hurtado et al. 2013]. Entrando nas metodologias ágeis Abad, Alipour e Ramsin (2012) propõem um *plug-in* para o EPFC, para permitir aos engenheiros de métodos construir metodologias ágeis através de uma abordagem de PME baseada em montagem. O *plug-in* fornece facilidades para a especificação das características de um determinado projeto, seleção de componentes de processo ágeis adequados do repositório de métodos e montagem final dos pedaços de método selecionados [Abad et al. 2012].

Três trabalhos abordam sobre implantação de melhorias e avaliação em processos de softwares, como Portela et al. (2014) apresentam um conjunto de ferramentas de suporte para a implementação de processos modelados no SPeM. Este conjunto de ferramentas, é chamada Spider-PE e tem como objetivo auxiliar as organizações de software na implementação do Modelo de Maturidade de Capacidades para Desenvolvimento (CMMI-DEV) e do Modelo de Referência de Melhoria de Processos de Software para Software (MR-MPS- SW) [Portela et al. 2014]. Hurtado, Bastarrica e Bergel (2013) apresentam uma ferramenta que permite analisar a qualidade dos modelos de processos de software SPeM 2.0. A ferramenta chamada Avispa, identifica uma série de padrões de erro e os destaca em planos diferentes. Uma descrição detalhada dos internos da Avispa é fornecida para mostrar sua estrutura e seus mecanismos de extensibilidade. Além disso, os autores apresentam um mecanismo interativo para definir novos *scripts* de análise e implementar novos padrões e planos [Hurtado Alegria et al. 2013a]. Já se tratando de avaliação, Gazel, Sezer e Tarhan (2012) apresentam uma ferramenta de avaliação de processo de software baseada em ontologia para coleta de dados da avaliação de processos e acompanhar a conformidade dos processos de software com o CMMI como modelo de

referência do processo [Gazel et al. 2012].

Muitos trabalhos selecionados abordam sobre adaptação de processos, definindo algumas variáveis para adaptar da melhor maneira o processo para determinado escopo, dentre esses trabalhos, Silvestre, Bastarrica e Ochoa 2014, apresentam uma ferramenta baseada em modelo que permite aos engenheiros de processos gerar automaticamente regras de transformação de adaptação usando uma interface gráfica de usuário [Silvestre et al. 2014]. Hurtado et al (2013) mostram uma abordagem baseada em modelo para a adaptação de processos de software, gerando automaticamente processos específicos do projeto com base no processo organizacional e nos contextos do projeto [Hurtado Alegria et al. 2013b]. Casare et al (2016) expõem Medusa, uma abordagem para adaptar processos de desenvolvimento de acordo com as necessidades do projeto. Ela se baseia em técnicas e métodos de engenharia de linha de produtos, leva em consideração as semelhanças e diferenças do processo, bem como fragmentos reutilizáveis [Casare et al. 2016]. Santos et al. (2015) relatam uma técnica que usa a mineração de processos para descobrir elementos do processo de software que são candidatos a adaptação. A abordagem analisa os *logs* de execução de várias instâncias de processo que compartilham um processo padrão comum [Santos et al. 2015]. Chaghrouchni, Kabbaj e Bakkoury (2014) apresentam uma nova abordagem para a evolução do processo, com base na adaptação dinâmica. Ao identificar os desvios do processo a adaptação dinâmica é usada para decidir a nova atividade ou conjunto de atividades a serem lançadas para reproduzir entradas e saídas solicitadas e para lidar com o impacto de desvio [Chaghrouchni et al. 2014].

Vários trabalhos estão relacionados com tratamento de variabilidades, como o de Rouillé et al. (2013) que propõe uma abordagem para resolução de variabilidades de processos em Linhas de Processos de Software durante a execução do processo [Rouillé et al. 2013]. Em outro trabalho Rouillé et al. (2012) relatam uma abordagem para definição de uma Linha de Processo de Software e a derivação automática de um processo a partir desta SPL de acordo com os requisitos de um determinado projeto. A variabilidade é gerenciada separadamente do modelo de processo utilizando uma linguagem de variabilidade comum [Rouillé et al. 2012]. Simmonds et al. (2012) apresentam uma combinação de notações e ferramentas para formalizar modelos de processos de software incluindo sua variabilidade, utilizando a ferramenta Modisco/AMW para garantir que somente a variabilidade razoável seja especificada [Simmonds et al. 2012]. Blum, Simmonds e Bastarrica (2015) propõem o algoritmo V que usa dois processos limiares para configurar uma Linha de Processo de Software. Relações muito frequentes são usadas para construir o processo base, as relações variáveis definem a variabilidade do processo e as relações raras são descartadas como ruídos [Blum et al. 2015]. Ayora et al. (2016) derivaram um conjunto de padrões de mudança para famílias de processos de construções de linguagem específicas de variabilidade identificadas na literatura, com objetivo de facilitar o gerenciamento de variabilidade nas famílias de processos, garantindo a correção familiar do processo e reduzir o esforço necessário para tais fins [Ayora et al. 2016].

4. Desenvolvimento da Ferramenta

Nesta seção é apresentado como foi construída a ferramenta para auxiliar na execução do processos. Na subseção 4.1 é descrito o processo de desenvolvimento e o que foi desenvolvido. Na subseção 4.2 é descrito por meio de diagramas UML como está a estrutura

de implementação do software.

4.1. Processo

O desenvolvimento do software foi guiado por algumas atividades propostas pelo SCRUM. Como a equipe atual é constituída de dois desenvolvedores, apenas algumas atividades foram seguidas. Abaixo uma lista das atividades propostas pelo SCRUM que utilizamos.

- Planejamento do Sprint
- Revisão do Sprint
- Execução do Sprint

Foi utilizado o conceito de Sprint e Sprint Backlog, no qual nos guiou em todo desenvolvimento. Definimos o tempo de cada Sprint em 2 semanas, no qual foi possível executar 2 Sprints até o desenvolvimento deste artigo. Na tabela 1 é apresentado as histórias de usuário contidas no Sprint Backlog e na tabela 2 é apresentado as histórias de usuário executadas, sendo a com ID 1 executada no primeiro *Sprint* e as demais no segundo.

Tabela 1. Sprint Backlog

| ID | História de Usuário | Prioridade |
|----|--|------------|
| 1 | Eu como usuário desejo poder selecionar um arquivo no formato SPEM e acompanhar a execução do processo. | 1 |
| 2 | Eu como usuário desejo poder finalizar cada atividade do processo com intuito de acompanhar o fluxo das atividades. | 2 |
| 3 | Eu como usuário desejo poder anexar arquivos nos artefatos gerados em cada uma das etapas de execução do meu processo. | 3 |
| 4 | Eu como usuário desejo poder gerar um arquivo contendo o software para execução do processo, a partir de um arquivo SPEM | 4 |

Tabela 2. Sprints 1 e 2

| ID | História de Usuário | Prioridade |
|----|---|------------|
| 1 | Eu como usuário desejo poder selecionar um arquivo no formato SPEM e ter como saída uma forma de acompanhar a execução do processo. | 1 |
| 2 | Eu como usuário desejo poder finalizar cada atividade do processo como intuito de acompanhar o fluxo das atividades. | 2 |
| 3 | Eu como usuário desejo poder anexar arquivos nos artefatos gerados em cada uma das etapas de execução do meu processo. | 3 |

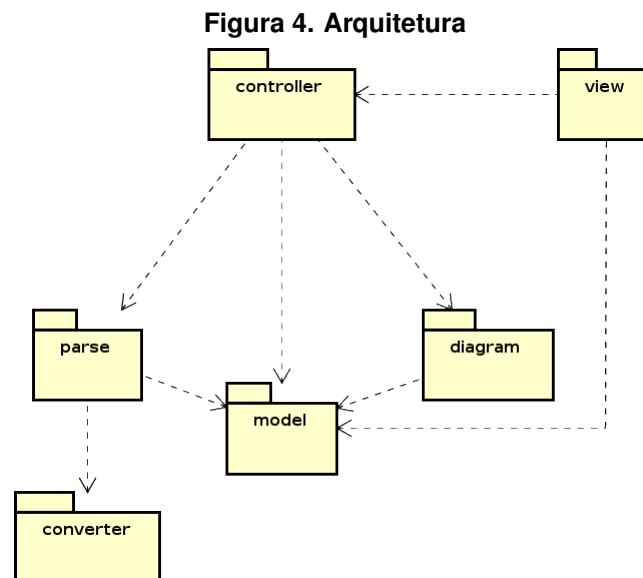
4.2. Implementação

O software foi criado utilizando a linguagem de programação JAVA com seus recursos da plataforma WEB. As principais tecnologias utilizadas para construção do software, foram: JSF, Primefaces, Bootstrap, XStream e Maven. O artigo não entrará em detalhes de implementação, apresentando apenas a relação entre as classes e a arquitetura do software.

Na subseção 4.2.1 é apresentado a arquitetura do software, descrevendo o objetivo de cada pacote de software. Na subseção 4.2.2 são apresentadas as relações entre as classes do software, descrevendo de forma simplificada a funcionalidade das principais classes envolvidas na geração do diagrama de classes.

4.2.1. Arquitetura

A figura 4 apresenta os principais pacotes envolvidos na geração da estrutura para controlar a execução de processos. Os pacotes *model* e *diagram* estão envolvidos com as regras de negócio do software. Os pacotes *parse* e *converter* estão envolvidos com a conversão do modelo SPEM representado em XML para objetos no Java em tempo de execução do software. O pacote *controller* tem como responsabilidade utilizar todos os demais pacotes oferecendo uma interface para o pacote *view* executar requisições do usuário.



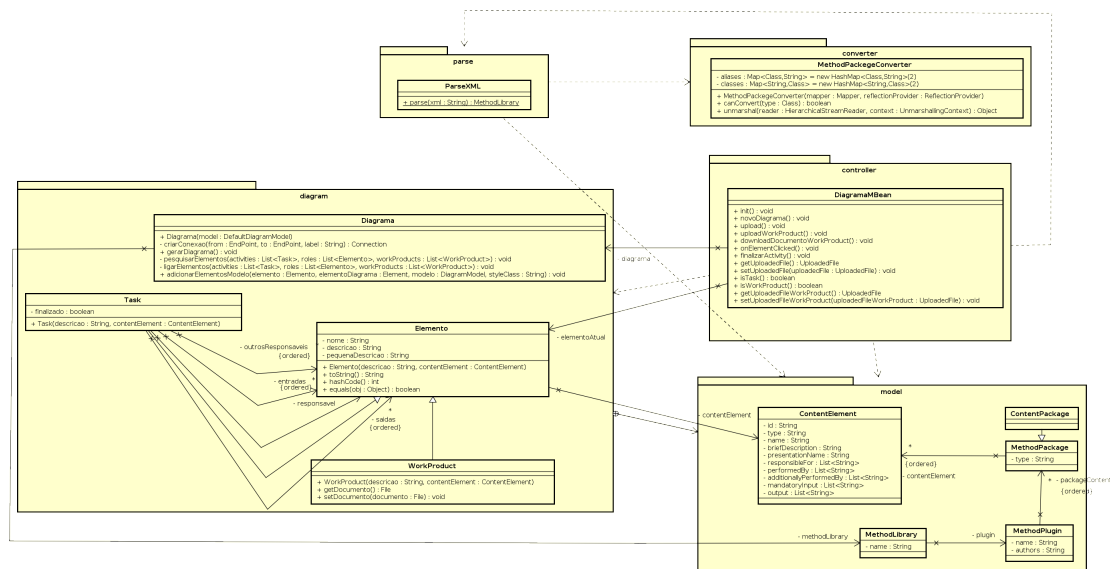
4.2.2. Diagrama de Classes

Na Figura 5 é apresentado o diagrama de classes, constituído das principais classes envolvidas na construção do software e suas relações.

A classe Diagrama do pacote *diagram* é utilizada para criar as conexões, pesquisar os elementos extraídos do arquivo SPEM e liga-los. A classe ParseXML é responsável pela extração dos elementos utilizados pela classe Diagrama. Esses elementos são extraídos seguindo a estrutura do arquivo SPEM gerado pela ferramenta EPF Composer. Essa estrutura está modelada no pacote *model*, onde as classes ContentElement, ContentPackage, MethodPackage, MethodLibrary e MethodPlugin representam as tags descritas no arquivo SPEM. As classes Task, WorkProduct e Elemento são a estrutura no qual utilizamos para gerar o diagrama, onde o diagrama pode ser constituído de um conjunto de objetos instanciados de cada uma dessas classes. Por fim a classe DiagramaMBean é encarregada de controlar as requisições do usuário feitas nas páginas. Essa classe utiliza a

tecnologia de gerenciamento de beans, no qual possibilita a ligação da interface com as páginas da visualização.

Figura 5. Diagrama de Classes



5. Resultados

A partir da implementação do software foi possível construir uma ferramenta de auxílio a execução de processos. O software apesar de simples possui uma estrutura robusta, possibilitada pela componentização que a linguagem Java oferece, podendo ter suas funcionalidades aumentadas ao longo do tempo.

Na figura 6 é apresentado a estrutura no qual o usuário seleciona o arquivo SPEM. O arquivo pode ser gerado a partir de um processo modelado na ferramenta EPF Composer.

Figura 6. Seleção Arquivo

Novo +

1. Escolha o arquivo SPEM

Selecionar arquivo...

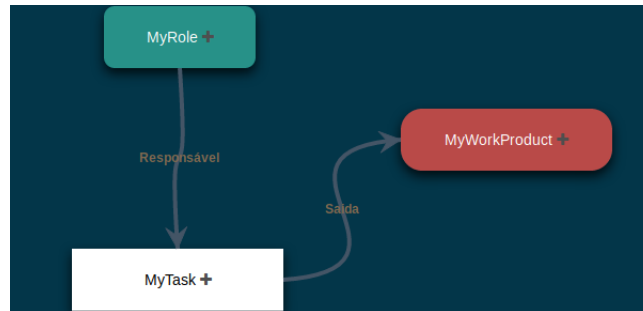
Nenhum arquivo selecionado.

2. Clique em enviar para gerar o diagrama

Enviar

Após o arquivo ser selecionado o usuário necessita apenas clicar no botão enviar. Com o arquivo no formato SPDM enviado para o servidor, a ferramenta vai gerar um diagrama como mostrado na figura 7. Este diagrama pode ser utilizado para controlar o fluxo das atividades, sendo possível finalizar cada atividade completada. Além disso é possível anexar arquivos a cada artefato da atividade, podendo ser posteriormente acessado novamente.

Figura 7. Diagrama Gerado



6. Conclusões

A partir do mapeamento sistemático conseguimos visualizar de forma mais abrangente as necessidades da área. Com essa visão levantamos a proposta de construir uma ferramenta para auxiliar uma das etapas do processo de software em um projeto. A etapa escolhida foi a execução do processo, no qual ocorre quando o processo é utilizado durante o desenvolvimento do software. Utilizando a tecnologia web oferecida pela linguagem Java alcançamos nosso objetivo, oferecendo uma ferramenta funcional para executar o processo.

Nos trabalhos futuros pretendemos oferecer uma estrutura do software baseada em linha de produto. Com isso conseguimos um software mais escalável, possibilitando por exemplo ao invés de anexar arquivos nos artefatos, criar esses artefatos diretamente no software.

Referências

- Abad, Z. S. H., Alipour, A., and Ramsin, R. (2012). Enhancing tool support for situational engineering of agile methodologies in eclipse. In *SERA (selected papers)*, pages 141–152. Springer.
- Ayora, C., Torres, V., de la Vara, J. L., and Pelechano, V. (2016). Variability management in process families through change patterns. *Information and Software Technology*, 74:86–104.
- Blum, F. R., Simmonds, J., and Bastarrica, M. C. (2015). Software process line discovery. In *Proceedings of the 2015 International Conference on Software and System Process*, pages 127–136. ACM.
- Casare, S., Ziadi, T., Brandão, A. A. F., and Guessoum, Z. (2016). Meduse: an approach for tailoring software development process. In *Engineering of Complex Computer Systems (ICECCS), 2016 21st International Conference on*, pages 197–200. IEEE.
- Chaghrouchni, T., Kabbaj, I. M., and Bakkoury, Z. (2014). Towards dynamic adaptation of the software process. In *Intelligent Systems: Theories and Applications (SITA-14), 2014 9th International Conference on*, pages 1–7. IEEE.
- Ellner, R., Al-Hilank, S., Jung, M., Kips, D., and Philippsen, M. (2012). An integrated tool chain for software process modeling and execution. In *Proceedings of the European Conference on Modelling Foundations and Applications*, pages 73–82.

- Gazel, S., Sezer, E. A., and Tarhan, A. (2012). An ontology based infrastructure to support cmmi-based software process assessment. *Gazi University Journal of Science*, 25(1):155–164.
- GENVIGIR, E. C., LUIZ FILHO, F., and SANT’ANNA, N. (2003). Modelagem de processos de software através do spem-software process engineering metamodel-conceitos e aplicação. *III WORCAP, São Jose dos Campos, SP*.
- Hurtado, J. A., Bastarrica, M. C., Ochoa, S. F., and Simmonds, J. (2013). Mde software process lines in small companies. *Journal of Systems and Software*, 86(5):1153–1171.
- Hurtado Alegría, J. A., Bastarrica, M. C., and Bergel, A. (2013a). Avispa: a tool for analyzing software process models. *Journal of Software: Evolution and Process*, 26(4):434–450.
- Hurtado Alegría, J. A., Bastarrica, M. C., Quispe, A., and Ochoa, S. F. (2013b). Mde-based process tailoring strategy. *Journal of Software: Evolution and Process*, 26(4):386–403.
- Kuhrmann, M., Kalus, G., and Then, M. (2014). The process enactment tool framework—transformation of software process models to prepare enactment. *Science of Computer Programming*, 79:172–188.
- Larman, C. (2007). *Utilizando UML e Padrões*. Bookman.
- Portela, C., Vasconcelos, A., Oliveira, S., Silva, A., and Elder, S. (2014). Spider-pe: A set of support tools to software process enactment. In *Proceedings of the 9th International Conference on Software Engineering Advances*.
- Rouillé, E., Combemale, B., Barais, O., Touzet, D., and Jézéquel, J.-M. (2012). Leveraging cvl to manage variability in software process lines. In *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific*, volume 1, pages 148–157. IEEE.
- Rouillé, E., Combemale, B., Barais, O., Touzet, D., and Jézéquel, J.-M. (2013). Integrating software process reuse and automation. In *Software Engineering Conference (APSEC), 2013 20th Asia-Pacific*, volume 1, pages 380–387. IEEE.
- Santos, R., Oliveira, T. C., et al. (2015). Mining software development process variations. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1657–1660. ACM.
- Silvestre, L., Bastarrica, M. C., and Ochoa, S. F. (2014). A model-based tool for generating software process model tailoring transformations. In *Model-Driven Engineering and Software Development (MODELSWARD), 2014 2nd International Conference on*, pages 533–540. IEEE.
- Simmonds, J., Bastarrica, M. C., Silvestre, L., and Quispe, A. (2012). Modeling variability in software process models. *Computer Science Department, Universidad de Chile, Santiago, Chile*.
- Sommerville, I. (2007). *Engenharia de software*, volume 8. Addison Wesley São Paulo.