7ª MOSTRA ACADÊMICA UNIMEP
Tema:
CIÊNCIA, TECNOLOGIA E INOVAÇÃO:
A universidade e a construção do futuro

7° Congresso de Pós-Graduação

REPRESENTAÇÃO DO CONHECIMENTO PARA MANUTENÇÃO DE DIAGRAMAS UML

Autor(es)

CLEVERTON FERREIRA BORBA

Orientador(es)

ANA ESTELA ANTUNES DA SILVA

1. Introdução

A fase de manutenção é uma das mais importantes na engenharia de software, esse conceito é afirmado por diversos autores, dentre eles podemos citar (PRESSMAN, 2005), (SOMERVILLE, 2003), (PADUELLI, 2007) e (PETERS, 2001).
Apesar do reconhecimento da importância da fase de manutenção de software, hainda dificuldades a serem transpostas tais como: custo, tempo e pessoal disponel para que o ciclo de vida do software se mantenha completamente documentado.
Quando uma solicitação de manutenção é feita, a manutenção do cigo se faz obrigatia para que a funcionalidade do software seja mantida. O problema é que a correção do cigo se torna, às vezes, a nica função da fase de manutenção. Porém, outras atividades do ciclo de vida deveriam também ser tratadas como parte do processo de manutenção. Diagramas desenvolvidos em fases anteriores não têm a atenção devida do desenvolvedor por inmeros motivos (sendo custo e tempo despendido os principais) causando uma série de problemas vindouros. Mesmo quando aplicada às fases de anise e projeto, a manutenção dos diagramas nem sempre é feita de maneira completa. Essa manutenção deveria contemplar todos os diagramas envolvidos, uma vez que, a mudança em um diagrama pode acarretar uma mudança em vios outros.
Este trabalho propõe a modelagem de uma base de conhecimento que represente a consistência existente entre os diagramas UML e a formalização, por meio de uma ontologia, das regras que regem a ligação entre os diagramas UML. A partir dessas representações, uma ferramenta pode ser desenvolvida para armazenar registros de manutenção.
A base deste artigo se faz sobre as seguinte bibliografias: (HNATKOWSKA, 2002), (HAUSMANN, 2002), (SAPNA, 2007), (ELAASAR, 2004), dentre outras.

2. Objetivos

1- Priorização à documentação de manutenção de diagramas UML; 2- Aplicação das fases de anise e projeto do modelo IEEE para a manutenção de diagramas UML; 3- Aplicação de ontologia de consistência entre diagramas UML para a manutenção dos mesmos; 4- Representação do conhecimento por meio de rede semântica e regras da ontologia.

3. Desenvolvimento

Segundo (PETERS, 2001), os produtos de software lançados não ficam em estado de congelamento, eles possuem erros e estão em constante manutenção. Corrigir esses erros detectados faz com que a aplicação original seja modificada e expandida. Se para a manutenção de linhas de cigo jexistem dificuldades de documentação, para manutenção de diagramas essas dificuldades são ainda maiores.

Segundo (SENGUPTA, 2008), a correção de erros de requisitos é mais dispendiosa do que a correção de erros de codificação, devido à atividade de re-projeto que pode ser necessia.
Segundo (PADUELLI, 2007), um dos problemas envolvidos na manutenção de software é a falta da manutenção da documentação existente envolvendo diagramas UML. Em (DANTAS, 2009) o autor afirma que a que a falta de manutenção dos diagramas UML deve-se à falta de ferramentas que auxiliem nessa tarefa.
A seguir uma taxonomia de manutenções é mencionada. Cinco tipos principais são destacados (IEEE, 1998), (PRESSMAN,2005), (SOMMERVILLE, 2003): • Manutenção Corretiva; • Manutenção Adaptativa; • Manutenção Perfectiva; • Manutenção Preventiva; • Manutenção Emergencial;
Qualquer que seja a classificação da manutenção para que a mesma seja realizada de forma correta, e que seja de sucesso, deve-se seguir um modelo, padrão, ou seja, um ciclo de vida apenas de manutenção. Nesta proposta, o modelo de manutenção do Instituto de Engenheiros Elétricos e Eletrônicos (do inglês, Institute of Electrical and Electronics Engineers - IEEE) foi adotado como paradigma.
O modelo do processo de manutenção preocupa-se em investigar as questões relativas a quem, o que, quando e como do processo sempre que ocorre uma solicitação de mudança. O modelo associa os mecanismos de entrada, saa e controle relativos a cada fase do processo de manutenção, conforme figura 1.
Esse modelo não pressupõe nenhum modelo de processo de software especico. A solicitação de mudança, a documentação de projeto, o cigo fonte, os bancos de dados e repositios são fontes de entrada. A vantagem desse modelo é que é aplicado ao planejamento de manutenção de software durante o desenvolvimento e à manutenção de sistemas legados. Para esta proposta, a manutenção dos diagramas UML seria aplicada às fases anise e projeto do ciclo de vida da manutenção IEEE.
Em termos de entrada, os seguintes dados são necessios: •tipo de diagrama atualizado: para essa proposta são considerados: diagrama de caso de uso, diagrama de classes, diagrama de sequência, diagrama de estados e diagrama de atividades; •solicitante: desenvolvedor que solicitou a manutenção; •sistema: nome do sistema ao qual o diagrama estassociado; •componente do diagrama alterado: descreve a parte do diagrama alterada, por exemplo, inclusão de ator em diagrama de caso de uso; •tipo de problema ocorrido: descreve a natureza do problema, por exemplo, inerente ao diagrama ou decorrente de alguma modificação do cigo etc.
Segundo (BOOCH, 2000), a modelagem é a parte central de todas as atividades que levam à implantação de um bom sistema. através dos diagramas desenvolvidos nas fases iniciais do ciclo de vida do software que o sucesso de todo o projeto é determinado.
A UML tornou-se uma das notações mais utilizadas para o desenvolvimento e modelagem de software. Segundo (ELAASAR, 2004), a principal vantagem da UML é de que se baseia na sua capacidade de expressar muitas visões de projeto, que vão desde a estrutural até a comportamental, utilizando um formalismo nico e integrado.
A vantagem de ser uma linguagem integrada gera o risco da correção de um diagrama e a não-correção de um diagrama interligado ao diagrama corrigido, gerando incompatibilidade de documentação. Por essa razão, a consistência entre diagramas tem sido investigada em diferentes domios (HAUSMANN, 2002), (SAPNA, 2007), (SCHREIBER, 1999), (SENGUPTA, 2008). Problemas de consistência revelam falhas de projeto.
A consistência, ou falta dela, tem as suas raes nos métodos formais (ELAASAR, 2004). Para afirmar que algo é consistente é necessio declarar o que é consistente com o objeto em questão. Como qualquer outra linguagem, a UML tem sua prria sintaxe e semântica. A correção sintica ou formal de um diagrama UML é geralmente uma condição prévia para qualquer nova anise de consistência.
Existem estudos na literatura que oferecem informações sobre a consistência na UML (HNATKOWSKA, 2002), (HAUSMANN, 2002), (KUZNIARZ, 2002) e a maioria dos trabalhos na ea concentra-se na necessidade de assegurar que essas transformações sejam consistência de preservação. Em (DEBONI, 2003) muitas informações sobre a consistência dos diagramas da UML podem ser encontradas.
Trabalhos que propõem modelos formais e semi-formais de consistência entre diagramas UML podem ser encontrados em (HA, 2008), (SAPNA, 2007), (SCHREIBER, 1999), (SENGUPTA, 2008), (ZAPATA, 2007). Para este trabalho foram analisados cinco dos diagramas da UML e a consistência entre eles. A tabela 1 foi gerada a partir de observações dos autores sobre a relação entre os diagramas UML (HA, 2008), (HNATKOWSKA, 2002), (HAUSMANN, 2002), (KUZNIARZ, 2002), (SAPNA, 2007), (SCHREIBER, 1999), (SENGUPTA, 2008), (ZAPATA, 2007).
A tabela 1 mostra algumas das principais consistências entre esses diagramas. Foram considerados os diagramas: caso de uso, classes, atividades, sequência e estados. Tabela 1. Principais consistências entre os diagramas UML.

4. Resultado e Discussão

Os principais conceitos sobre o conhecimento necessio para realizar consistências entre os diagramas UML foram retirados da Tabela 1. Para a representação do conhecimento foram utilizadas uma rede semântica e uma ontologia de regras de consistência baseada em uma gramica livre de contexto.
Rede Semântica é uma das mais simples e utilizadas técnicas de representação conceitual de domios de conhecimento (GELLER,

2002). Cada conceito do domio de conhecimento foi transformado em um nda rede semântica apresentada na figura 1. Figura 1. Rede Semântica sobre Ligações entre Diagramas UML.
Cada npode estar relacionado semanticamente a vios outros conceitos de outros diagramas. Tais relacionamentos indicam que, em caso de manutenção em algum desses componentes, os componentes a ele relacionados podem ter que ser modificados também.
Ontologia de consistência entre diagramas UML Em (BORST, 2006) a seguinte definição de ontologias é encontrada: "Uma ontologia é uma especificação formal e explita de uma conceitualização compartilhada". A palavra, “formal” significa traduzel para computadores; "especificação explita" diz respeito a conceitos, propriedades, relações, funções, restrições, axiomas, explicitamente definidos; "compartilhado" quer dizer conhecimento consensual e "conceitualização" diz respeito a um modelo abstrato de algum fenômeno do mundo real.
A conceitualização é formada por um vocabulio controlado que é arranjado hierarquicamente e através de relações entre conceitos, como nas taxonomias e tesauros. Uma conceitualização é uma visão abstrata e simplificada do mundo que se deseja representar (HINZ, 2006).
Para a formalização da ontologia de consistência entre os diagramas UML serutilizado o formalismo de linguagens baseadas em Gramicas Livres de Contexto (COHEN, 1986). Uma gramica livre de contexto G é uma qurupla (V, ∑, R, S) sendo: V: um alfabeto; ∑: conjunto de terminais, tal que ∑ ⊂ V; R: conjunto de regras e S: sbolo inicial da gramica.
A figura 3 mostra uma gramica livre de contexto para consistências entre diagramas UML. As regras foram baseadas na rede semântica da figura 1. Figura 2.
Parte de uma gramica livre de contexto para consistência entre diagramas UML. A rede semântica da seção 4.1 e as regras da seção 4.2 representam a base de conhecimento do domio de consistências entre diagramas UML. Essa base deverser formada pelos registros de manutenção e pelas regras de consistência da gramica livre de contexto. A entrada de um registro de manutenção serdada por: •tipo de diagrama atualizado: para essa proposta são considerados: diagrama de caso de uso, diagrama de classes, diagrama de sequência, diagrama de estados e diagrama de atividades; •solicitante: desenvolvedor que solicitou a manutenção; •sistema: nome do sistema ao qual o diagrama estassociado; •componente do diagrama alterado: descreve a parte do diagrama alterada, por exemplo, inclusão de ator em diagrama de caso de uso; •tipo de problema ocorrido: descreve a natureza do problema, por exemplo, inerente ao diagrama ou decorrente de alguma modificação do cigo etc.
Essas informações serão armazenadas na base de conhecimento. Essas instâncias são instâncias do domio do conhecimento, ou seja, instâncias do especialista e não instâncias de manutenções (SCHREIBER, 1999). As instâncias da ontologia serão utilizadas para a realização da consistência de ocorrências de manutenção entre os diagramas UML.
Como trabalho futuro pretende-se implementar a base de conhecimento de instâncias de conhecimento e instâncias de ocorrências em uma ferramenta para armazenamento de manutenções de diagramas UML. 5. Trabalhos Futuros A necessidade dos profissionais em assegurar que a manutenção dos diagramas da UML seja realizada com sucesso motiva o desenvolvimento de uma ferramenta que auxilie a correção e armazenamento dos mesmos.
A ferramenta tem como principal objetivo o apontamento de correções necessias entre diagramas da UML a partir da necessidade de correção em determinado diagrama.
possel verificar que sercriada uma base de conhecimento na qual serão cadastradas as manutenções de diagramas UML de anise e projeto existentes na empresa. Para cada manutenção, a verificação dos diagramas se faz necessia e a consistência entre eles é realizada pelo conjunto de regras previamente cadastradas na base de conhecimento.
As manutenções cadastradas na ferramenta formarão uma base de conhecimento que auxiliaros profissionais do projeto de software a administrar e corrigir posseis problemas na documentação dos diagramas UML.

5. Considerações Finais

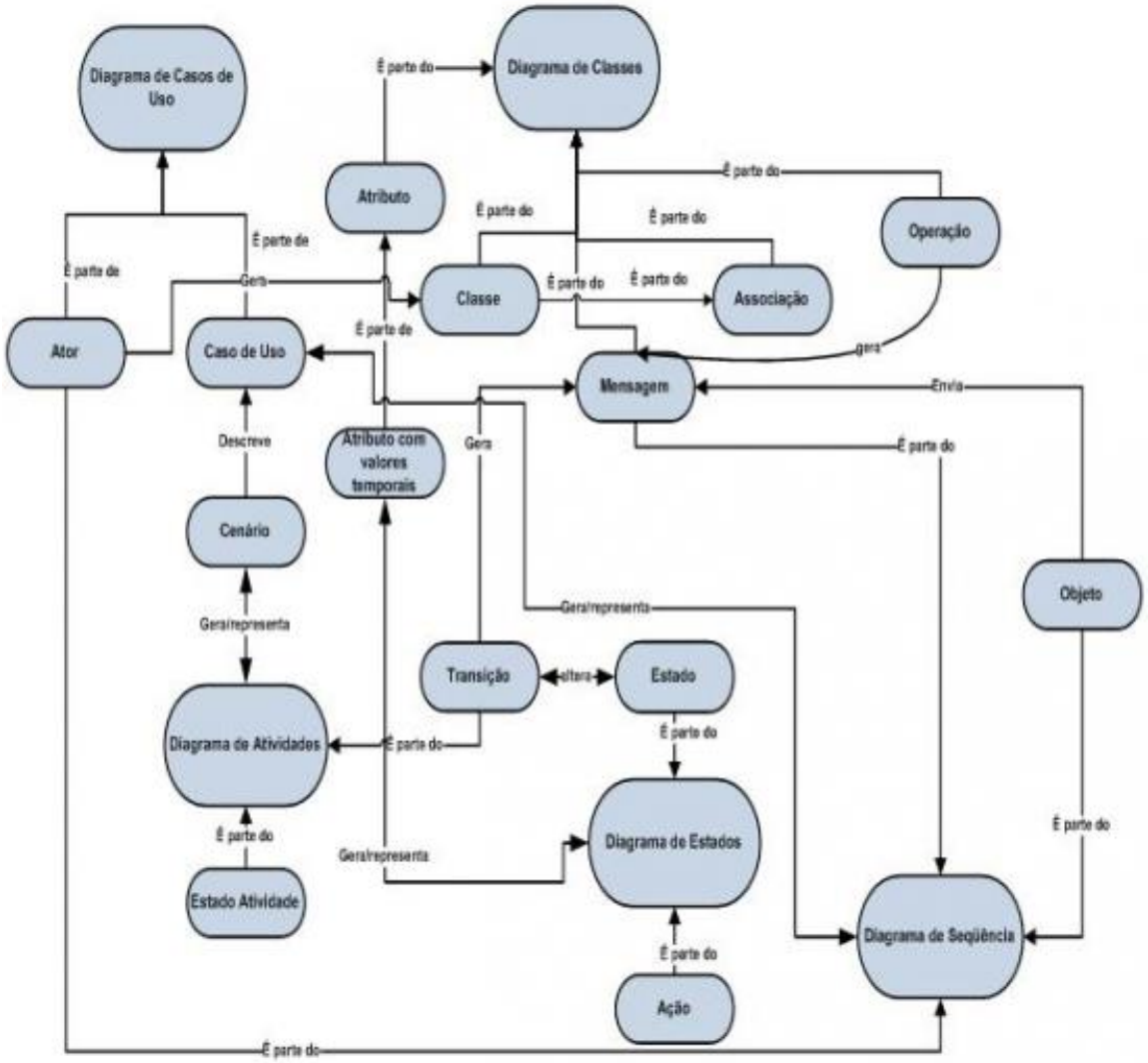
Apesar do reconhecimento da importância da fase de manutenção de software, poucos analistas e gerentes tem se dedicado à correção da documentação essencial para que o ciclo de vida do software se mantenha completamente documentado.
Dessa forma, esse trabalho traz como contribuições:1- Priorização à documentação de manutenção de diagramas UML; 2- Aplicação das fases de anise e projeto do modelo IEEE para a manutenção de diagramas UML; 3- Representação de uma base de conhecimento para consistência entre diagramas UML baseada em gramica livre de contexto e redes semânticas.
Como trabalhos futuros imediatos são citados: o aprimoramento da ontologia de consistência e a criação da ferramenta de acordo com a metodologia apresentada nesse trabalho. Como trabalho posterior, criação de algoritmo que possa minerar a base de instâncias de manutenções para detecção de padrões nas solicitações de manutenções de diagramas UML.

Referências Bibliográficas

BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar. UML – guia do usuio. 10ª ed. Rio de Janeiro: Editora Campus, 2000.
BORST, W. N. Construction of engineering ontologies. Tese (Doutorado). 60 Disponel em: http://www.ub.utwente.nl/webdocs/inf/1/t0000004.pdf, 2006
COHEN, D. Introduction to Computer Theory. John Wiley & Sons, 1986.
CORAZZON, R. What is ontology?. Disponel em: http://www.formalontology.it/ section_4.htm, 2009. DANTAS, C., MURTA, L., WERNER, C. Odyssey-WI: Uma Ferramenta para mineração de Rastros de Modificação em Modelos UML Versionados. Disponel em:http://reuse.cos.ufrj.br/prometeus/publicacoes/TS%20Odyssey-WI%20Final.pdf, 2009.
DEBONI, José Eduardo Zindel. Modelagem orientanda a objetos com a UML. São Paulo: Futura, 2003.
ELAASAR, M. BRIAND, L. An Overview of UML Consitency Management. Department of Systems and Computer Engineering, Ontatio, Canada, 2004.
GELLER, J.; Perl, Y.; HALPER, M.; CHEN, Z.; Gu, H. Evaluation and Application of Semantic Networks Partition. IEEE Transactions on Information Technology in Biomedicine. Vol. 6, n.2, pp. 109-115, 2002
HA, IlKyu; KANG,Byunguk. Cross Checking Rules to Improve Consistency between UML Static Diagram and Dynamic Diagram. C. Fyfe et al. (Eds.): LNCS 5326, pp. 436–443. Springer-Verlag Berlin Heidelberg, 2008.
HNATKOWSKA, B., HUZAR, Z., KUZNIARZ, L., TUZINKIEWICZ, L. A systematic approach to consistency within UML based software development process. Available at (KUZNIARZ ,2002), page 16-29, 2002
HAUSMANN, J., HECKEL, R., SAUER, S. Extended Model Relations with Graphical Consistency Conditions. Disponel em (Kuzniarz, 2002), p. 61-74, 2002
HINZ, V. Proposta de Criação de uma Ontologia de Ontologias. Universidade Catica de Pelotas, Escola de Informica. Pelotas, Brasil, 2006.
IEEE. Standard for Software Maintenance. Software Engineering Standards Committee of the IEEE Computer Society, 1998.
KUZNIARZ, L., REGGIO, G., SOURROUILLE, J. HUZAR., Z. Workshop on Consistency problems in UML-based Software Development. UML 2002. Blekinge Institute of Technology, 2002
PADUELLI, M. M. Manutenção de Software: problemas ticos e diretrizes para uma disciplina especica. USP – São Carlos, 2007
PETERS, James F. Engenharia de Software; tradução de Ana Patria Garcia. Rio de Janeiro: Campus, 2001
PRESSMAN, Roger S. Engenharia de Software. Tradução José Carlos Barbosa dos Santos. São Paulo: Person Education do Brasil, 2005.
SAPNA, P.G.; MOHANTY, H. Ensuring Consistency in Relational Repository of UML Models. 10th International Conference on Information Technology. IEEE Computer Society, pp. 217-222, 2007.
SCHREIBER, G. Knowledge Engineering and Management. Cambridge. MIT Press, 1999.
SENGUPTA, S.; BHATTACHARYA, S. Formalization of UML Diagrams and Their Consistency Verification —A Z Notation Based Approach. ISEC'08, pp. 151-152. Hyderabad, India, 2008.
SOMMERVILLE, Ian. Engenharia de Software; tradução André Maurio de Andrade Ribeiro. São Paulo: Addison Wesley, 2003
ZAPATA, C. M.; GONZEZ, G.; GELBUKH, A. A Rule-Based System for Assessing Consistency Between UML Models. A. Gelbukh and A.F. Kuri Morales (Eds.):LNAI 4827, pp. 215–224. Springer-Verlag Berlin Heidelberg, 2007.

Anexos

	Diagrama de Caso de Uso	Diagrama de Classes	Diagrama de Seqüência	Diagrama de Estados	Diagrama de Atividades
Diagrama de Caso de Uso	-	Todos os atores indicados nos casos de uso são candidatos a classes no diagrama de classes	É necessário que todos os cenários representados pelos diagramas de seqüência estejam presentes no diagrama de caso de uso	Não existe ligação direta, mas sim indireta passando pelo diagrama de classes.	É uma alternativa para representar um processo descrito por um caso de uso.
Diagrama de Classes	Todos os atores são representados em classes. Logo para qualquer modificação na classe correspondente, seria necessária uma revisão no diagrama de caso de uso.	-	Os diagramas de classe e seqüência estão diretamente ligados exatamente porque o diagrama de seqüência é a demonstração das mensagens trocadas entre as classes no tempo de execução do *software*.	O diagrama de estados representa a dinâmica interna dos atributos da classe. Ele expõe como os atributos reagem aos estímulos externos que recebe.	Cada cenário de execução de uma operação pode ser mostrado por um diagrama de atividade
Diagrama de Seqüência	Diagrama totalmente dependente do diagrama de caso de uso. Qualquer modificação no original afeta diretamente o outro.	Qualquer modificação do diagrama de seqüência, indica que alguma mensagem entre as classes deve ser alterada.	-	Sem ligação direta	Cada cenário de execução de uma operação pode ser mostrado por um diagrama de atividade
Diagrama de Estados	Sem ligação direta	Qualquer modificação do diagrama de estados indica que a dinâmica interna da classe foi alterada. O que leva a uma análise do diagrama de classes.	Sem ligação direta	-	Pode alterar a transição de alguma atividade
Diagrama de Atividades	Sendo uma alternativa para representar os diagramas de casos de uso, a ligação com os mesmos é direta em caso de alteração.	Como o diagrama de atividades combina mensagens internas e externas da classe, qualquer modificação sua leva à análise do diagrama de classes.	Como um diagrama de atividade pode representar o cenário de uma operação a mudança em um deles pode representar a mudança na seqüência do diagrama de seqüência correspondente	Sem ligação direta	-



$\sum_{conceitos}$ = {ação, associação, atividade, ator, atributo, atrib_altem, caso_uso, cenário, classe, estado, estado_atividade, mensagem, objeto, operação, transição}
$\sum_{relacionamentos}$ = {altera, descreve, gera, representa, envia}

Regra1. S →DiagramaConsistênciaDiagrama
Regra2. Diagrama→diagclasses|diagcaso_uso|diagatividades|diagestados|diagsequência
Regra3. Diagrama → ComponenteDiagrama
Regra4. ComponenteDiagrama → ação|associação|atividade|ator|atributo|atrib_altem|caso_uso|cenário|classe|estado|estado_atividade|mensagem|objeto|operação|transição
Regra5. Consistência→ConsistênciaCaso_Uso|ConsistênciaAtividades|ConsistênciaEstados|ConsistênciaSequência|ConsistênciaClasses
Regra6. ConsistênciaCaso_Uso→gera|descreve
Regra7. ConsistênciaClasses→gera
Regra8. ConsistênciaAtividades→representa|gera|altera
Regra9. ConsistênciaEstados→ altera|representa
Regra10. ConsistênciaSequência→representa|envia