UM DIAGNÓSTICO DO DESENVOLVIMENTO DE SOFTWARE DE UMA MPE

A DIAGNOSIS OF SOFTWARE DEVELOPMENT IN A MPE

Anderson dos Santos Lima[1]
Gislaine Camila Lapasini Leal[2]

Resumo: O Desenvolvimento de Software se tornou uma atividade chave e crítica para muitas empresas. Um produto de software com qualidade requer um processo de desenvolvimento também com qualidade. Para isso, metodologias de desenvolvimento podem ser empregadas, como o RUP, o XP ou o AUP. A complexidade de algumas metodologias, entretanto, requer que elas sejam adaptadas à realidade da organização. A microempresa objeto deste artigo não dispunha de um processo de desenvolvimento formalizado, o que acarretava muitos problemas como estouro nos prazos, bugs constantes e insatisfação por parte dos clientes. O objetivo deste trabalho foi formalizar um processo de desenvolvimento, pautado em uma metodologia dentre as exploradas. O RUP foi escolhido como a base para o processo da empresa, sendo adaptado com os princípios ágeis do XP. O principal objetivo ao se formalizar um processo é garantir maior qualidade no produto final, ainda que muitas vezes esse objetivo não seja alcançado. A implantação de um modelo de desenvolvimento requer, dentre outras coisas, comprometimento de toda equipe. A principal dificuldade consiste em convencer as pessoas que o maior tempo deve estar pautado no planejamento das atividades. O desenvolvimento ágil colabora para que somente atividades que agreguem valor ao produto sejam executadas, diminuindo, dessa maneira, tempo de desenvolvimento desnecessário.

Palavras-Chave: Desenvolvimento de Software. ISO/IEC 12207. RUP – Rational Unified Process. XP – Extreme Programming. AUP - Agile Unified Process. MPS-BR. CMMI. Desenvolvimento Ágil. Melhoria Contínua.

Abstract: The software development has become a key and critical activity for many companies. A software product with quality requires a development process also with quality. Thereunto, development methodologies can be used/applied, such as RUP, XP or AUP. The complexity of some methods, however, requires them to be adapted to the organization's reality. The target company of this article did not have a formalized development process, which caused many troubles such as term overflow, bugs and constant customer's dissatisfaction. The objective of this study was to formalize a development process, based on a methodology among the exploited. The RUP was chosen as the business process base,

[1] Acadêmico do Curso de Engenharia de Produção da Universidade Estadual de Maringá
andwarf2004@hotmail.com
[2] Professora orientadora – Engenharia de Produção – Universidade Estadual de Maringá
gclleal@uem.br
Universidade Estadual de Maringá
Avenida Colombo, 5790, Bloco 19/20, Sala 05
CEP: 97020-900, Maringá, Paraná.

being adapted to the agile principles of XP. The main reason to formalize a process is to improve final product's quality. A development model implementation requires, among other things, the involvement of the whole team. The main difficulty is to convince people that most of the time must be used in the activities planning. Agile development collaborates with only value-added activities to run, reducing in this way, unnecessary development time.

Keywords: Software Development. ISO/IEC 12207. RUP – Rational Unified Process. XP – Extreme Programming. AUP - Agile Unified Process. MPS-BR. CMMI. Agile Development. Continuous Improvement.

1 INTRODUÇÃO

A demanda crescente por softwares nas mais variadas áreas tem feito com que as empresas se preocupem cada vez mais com o desenvolvimento de software com qualidade, e utilizando as novas tecnologias, a melhoria de processos, a capacitação dos colaboradores, dentre outros meios, elas conseguem alcançar este objetivo (Nascimento, 2008).

Nos últimos anos, tem sido observado um aumento considerável de empresas brasileiras envolvidas em programas de melhoria de processo de software. Estes programas têm sido planejados e executados pela necessidade de melhoria no desenvolvimento de software e pela demanda crescente no mercado por empresas reconhecidas em níveis de maturidade nos principais modelos existentes no mercado (Hilgert et al., 2008).

Um sistema de qualidade requer um processo de desenvolvimento também de qualidade. Para isso, são necessários diversos esforços para minimização e correção dos erros que eventualmente possam surgir.

Este artigo propõe um processo de desenvolvimento de software para uma microempresa, que atua no ramo de sistemas voltados para provedores de internet. Inicialmente foi realizado um diagnóstico dos principais problemas relacionados ao desenvolvimento de software na empresa em questão e, em seguida, um novo processo foi estruturado.

2 MÉTODO

Do ponto de vista de sua natureza, a pesquisa é considerada do tipo Aplicada. Em relação aos objetivos é Explicativa. No que se refere aos procedimentos técnicos, é considerada uma Pesquisa-Ação, pois está diretamente relacionada com a solução de um problema específico e, além disso, os representantes da situação estão envolvidos de algum modo (Silva e Menezes, 2005).

A execução da pesquisa foi baseada em observações in loco, entrevistas com a Equipe de Desenvolvimento e análise de documentos que a Empresa possui.

3 PROCESSO ATUAL E DIAGNÓSTICO DE PROBLEMAS

O processo de desenvolvimento da empresa não está formalizado, mas existe. Por mais abstrato que seja, existem etapas e tarefas que são desenvolvidas seguindo uma sequência lógica. O desenvolvimento pode ser por dois modos: Implementação de novos módulos e Manutenção em módulos existentes.

Atualmente a empresa lista os principais recursos que serão desenvolvidos no sistema em rascunhos de papel. Ao final de um determinado ciclo de desenvolvimento, é elaborado um documento chamado Changelog, que contém todos os recursos que foram desenvolvidos, bem como aqueles que foram corrigidos e alterados. O objetivo do Changelog é dar apoio aos usuários do sistema sobre tudo que foi feito na nova versão. O Changelog é elaborado pelos próprios desenvolvedores.

A falta de um processo formalizado acarreta muitos problemas e dificuldades em todas as fases do desenvolvimento, o que impacta sobre a produtividade. O primeiro problema identificado, são os erros que surgem quando um módulo [1]que depende de outros é alterado. Este problema decorre da falta de documentação. Atualmente o sistema possui muitos módulos e muitas aplicações – por exemplo, módulos financeiro, fiscal e contábil -, grande parte interligados e que se comunicam constantemente (alto acoplamento). Durante a manutenção, os módulos dependentes estão apenas no conhecimento do desenvolvedor, pois não há nenhum documento relacionando todas as dependências que existem (por exemplo, diagrama de entidade-relacionamento). Assim, não é raro que aplicações necessárias sejam deixadas de lado durante a manutenção de determinado módulo – por exemplo, ao se alterar determinados códigos no módulo financeiro que impactarão diretamente o módulo fiscal. Conforme relato do principal desenvolvedor da equipe, em determinada situação, uma simples adição de um campo em determinada tabela do banco de dados foi responsável pela "parada" de dois módulos

[1] Um módulo consiste num conjunto de códigos fonte específicos.

financeiros muito importantes. Este problema não teria ocorrido se houvesse uma documentação específica sobre aquele módulo.

O segundo problema é o estouro nos prazos estabelecidos. O negócio da empresa não consiste em determinar um prazo fechado em que novas atualizações serão lançadas. Contudo, a empresa tem em sua política que o sistema não deve ficar mais de um mês sem atualização. Em contrapartida, geralmente o tempo para lançamento de uma atualização pode chegar a quatro vezes o estipulado. Isso gera muita insatisfação por parte dos clientes. O lançamento da versão 2.0.000 do sistema ocorreu no dia 26 de agosto de 2010. Um release foi lançado nos dias seguintes, para corrigir bugs inesperados que foram encontrados. Após este release, a próxima versão do sistema demorou em torno de quatro meses para ser lançada.

Outro problema identificado é a falta de priorização entre as novas implementações e as manutenções constantes. Várias vezes a equipe de desenvolvimento para sua atividade para dar suporte e manutenção ao problema de determinado cliente, o que contribui ainda mais para os estouros dos prazos.

A falta de documentação também é um problema grave. Os módulos do sistema possuem alto grau de acoplamento. No início do desenvolvimento perde-se muito tempo identificando todas as relações entre módulos. Além disso, problemas e dúvidas "comuns" – ou seja, que já foram identificados e resolvidos em outros momentos – requerem o mesmo esforço para encontrar uma solução. Estas duas situações seriam facilmente resolvidas se houvesse uma documentação mínima.

Pode-se apontar ainda a falta de gerenciamento de testes adequado. Atualmente os poucos testes realizados não são documentados e muitas vezes cenários importantes deixam de ser testados, fazendo com que bugs não sejam identificados enquanto a nova versão ainda não foi lançada. A Figura 1 apresenta a quantidade de erros corrigidos nas últimas versões do sistema. Pode-se constatar que as atualizações lançadas com maior atraso – versões 2.0.002 e 2.0.006 respectivamente – são as que possuem maior quantidade de erros a serem corrigidos. A contagem foi realizada através das Cartas de Atualização, uma vez que são a única documentação disponível no momento.

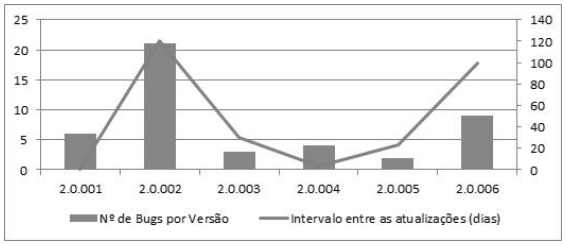

Figura 1. Gráfico do nº de bugs por versão

Por fim, a falta de um processo formalizado não possibilita o acompanhamento e avaliação da produtividade da Equipe de Desenvolvimento.

4 PROCESSO DE DESENVOLVIMENTO PROPOSTO

A estruturação de um Processo de Desenvolvimento procurou deixar toda e qualquer informalidade de lado, para que a solução de problemas seja realiza de maneira sistemática. Dessa maneira, o processo foi pautado no registro das ações e resultados das mesmas, como forma de análise e avaliação do processo como um todo.

O início deu-se com o contato entre a Equipe de Desenvolvimento e os responsáveis pelo setor, uma vez que a gerência de nível superior tem que estar a par de tudo o que está acontecendo. O planejamento foi mostrado como um dos principais meios através do qual o setor de desenvolvimento conseguirá atingir seus objetivos de maneira eficiente, eficaz e com qualidade. Por este motivo, a própria gerência chegou à conclusão que tempo gasto com planejamento, é tempo economizado com manutenções futuras.

A estruturação do Processo de Desenvolvimento foi realizada tomando como base a metodologia RUP e agregando práticas da metodologia XP, bem como se pautando nos seguintes processos da norma ISO/IEC 12207:

- Desenvolvimento (Processos Fundamentais);
- Documentação e Verificação (Processos de Apoio);
- Melhoria Contínua (Processos Organizacionais).

Este processo é totalmente focado no projeto de novas versões de um sistema já existente, o que engloba a adição de novos recursos, manutenção e melhorias no mesmo.

O modelo proposto levou em consideração algumas práticas de desenvolvimento do RUP, a saber:

- O desenvolvimento do software será iterativo e incremental, através de um ciclo de vida pautado em disciplinas que delineiam as diretrizes do desenvolvimento;

- O gerenciamento de requisitos será realizado por um papel específico, e constituirá a primeira atividade no desenvolvimento de um novo release;
- A modelagem visual será utilizada na forma de modelos UML, principalmente diagramas de casos de uso, contribuindo na compreensão do sistema e na comunicação entre quem está passando os requisitos e quem está recebendo;
- A verificação contínua da qualidade se dará através de planejamento de testes, revisões e verificações durante todo o desenvolvimento e não apenas ao final deste.

Os papéis têm como objetivo definir responsabilidades e deveres. São em número de 10, sendo eles:

- Analista de Requisitos: é o responsável pela definição dos recursos que o release deverá possuir e opcionalmente, da maneira com que a solução será implementada;
- Gerente de Projetos: é o responsável pela delimitação dos prazos para conclusão do release;
- Programador: constitui o elemento principal da Equipe de Desenvolvimento. Seu objetivo primário é a implementação das aplicações;
- Programador Chefe: é o responsável pela Equipe de Desenvolvimento. Além de desenvolver, ele gerencia a equipe, avalia os resultados e define diretrizes;
- Projetista de Banco de Dados: é o responsável por definir a estrutura final do banco de dados;
- Programador de Testes: é o responsável pela criação de modelos de testes para serem utilizados nas atividades de teste pela Equipe de Desenvolvimento;
- Projetista de Interfaces: é o responsável por criar a interface de interação entre a aplicação e o usuário da mesma;
- Redator Técnico: é o responsável pela criação da documentação do release para o usuário final;
- Revisor de Documentação: é o responsável pela revisão de documentos voltados aos usuários do sistema (geralmente os documentos com as novidades do release e manuais do usuário);
- Tradutor: é o responsável por traduzir os documentos voltados aos usuários do sistema (geralmente os documentos com as novidades do release e manuais do usuário) para outras línguas que forem necessárias.

O processo prevê um ciclo de vida pautado em disciplinas que delineiam as diretrizes do desenvolvimento. As disciplinas são importantes para dividir as atividades e tarefas e também proporcionar um controle melhor de cada artefato e de cada papel. A seguir, as 8 disciplinas são detalhadas.

4.1 GERÊNCIA DE PROJETOS

O principal objetivo desta disciplina é gerenciar as metas conforme os prazos estabelecidos. O gerenciamento dos projetos requer que todas as tarefas do mesmo sejam especificadas, possuam prazos iniciais e finais, tenham marcos definidos e sejam passíveis de avaliação, pela Equipe de Desenvolvimento e pela Gerência. A disciplina é constituída pelas seguintes atividades:

- Avaliar projetos anteriores, através dos quais será possível realizar uma estimativa mais próxima da realidade;
- Avaliar o Plano de Desenvolvimento, no qual estarão elencados todos os requisitos do novo release;
- Definir tarefas necessárias para cada membro da equipe;
- Estimar um prazo de realização para cada tarefa;
- Elaborar cronograma, geralmente através de uma ferramenta específica, como o Microsoft Project, por exemplo. Algumas empresas – como a do estudo em questão – possuem um gerenciador de projetos próprio, através do qual fazem a programação das atividades; Divulgar e Distribuir cronograma a todos os membros da equipe.

4.2 REQUISITOS

A disciplina Requisitos seleciona, avalia e define os requisitos do sistema, principalmente no que tange às especificações do cliente. As atividades desta disciplina são realizadas em forma de reuniões entre as pessoas envolvidas, para que todas as dúvidas sejam sanadas e não haja diferentes entendimentos de um mesmo requisito. A disciplina é constituída pelas seguintes atividades:

- Definir todos os itens que farão parte do release, sejam novos recursos, correções de erros ou alterações em aplicações existentes, sendo apresentada uma visão superficial sobre cada item.

- Detalhar cada item apontado, explicando a importância do mesmo, o motivo dele estar sendo incluído no release e as principais alterações que isto causará no sistema como um todo. Muitas vezes esta atividade não conterá todos os itens do release, sendo necessário realizá-la novamente posteriormente, quantas vezes forem necessárias, até que todos os itens sejam completamente definidos;
- Sugerir e propor melhorias;
- Elaborar Plano de Desenvolvimento;
- Criar ou Atualizar Diagramas de Casos de Uso. A atualização ocorrerá quando o item elencado for uma adição ou alteração em um recurso/aplicação já existente no sistema, que acarrete uma mudança significativa no mesmo.
- Avaliar Diagramas de Casos de Uso.

4.3 MODELAGEM DE NEGÓCIO

Esta disciplina tem o objetivo de entender as regras de negócio das novas aplicações do release. É muito comum que, para se desenvolver alguma aplicação, sejam necessários estudos aprofundados em determinadas áreas, como contabilidade, finanças, redes, etc. A disciplina agrega novos conhecimentos e dá suporte ao desenvolvimento de uma maneira eficiente. A falta deste conhecimento pode provocar inconsistências e, em um estágio mais avançado do desenvolvimento, problemas que deverão ser solucionados a um custo muito maior do que se tivessem sido evitados no início. A disciplina é constituída pelas seguintes atividades:

- Avaliar requisitos que necessitem de um estudo aprofundado sobre determinado assunto;
- Identificar materiais necessários – geralmente manuais, apostilas, artigos, etc. – que darão suporte ao entendimento de todas as regras de negócio necessárias;
- Estimar tempo necessário para entendimento das regras de negócio;
- Estudar materiais identificados. Esta atividade pode durar bastante tempo, dependendo do nível de estudo requerido ou da complexidade do assunto;
- Criar documento específico, quando necessário. A criação deste documento depende da regra de negócio que está sendo estudada e não é obrigatória. Uma tabela relacionando intervalos de valores por atributos – para nota fiscal eletrônica, por exemplo – é um exemplo de documento que pode ser gerado. Há casos em que não é gerado documento algum.

4.4 ANÁLISE E PROJETO

Recursos aparentemente simples, como a adição de um novo campo para guardar alguma informação ou o repasse de informações para outra aplicação, podem desencadear uma séria de complicações conceituais e estruturais em um sistema. A disciplina de Análise e Projeto visa encontrar soluções para todos os requisitos que farão parte do release.

Em um sistema relativamente grande, a quantidade de aplicações, ligações e dependências entre elas pode se tornar um quebra-cabeça muito complicado de se resolver. Por isso, a Análise e Projeto avalia o sistema da maneira como está hoje e como estará futuramente, levando em consideração aspectos como número de clientes e número de informações armazenadas pelo banco de dados. A disciplina é constituída pelas seguintes atividades:

- Analisar Diagramas de Casos de Uso e documentos referentes às Regras de Negócio;
- Criar Diagrama de Entidade-Relacionamento. Esta atividade pode ser realizada com a ajuda de um software como o Microsoft Visio;
- Projetar Banco de Dados;
- Analisar requisitos de interface;
- Projetar Interface. Esta modelagem geralmente é realizada à mão, em uma folha de papel, conforme os requisitos do release;
- Analisar dependências entre módulos existentes. O projeto de um release muitas vezes demanda o incremento de funcionalidades em aplicações já existentes, por isto esta análise é fundamental;
- Analisar alterações de Banco de Dados e Interface. Muitas vezes novos recursos requerem alterações importantes em aplicações já existentes, como a criação de novos campos, alteração do banco de dados ou mesmo alterações de layout;
- Definir aplicações que sofrerão alterações e de quais tipos;
- Definir todas as alterações identificadas nas duas atividades anteriores. Esta atividade é realizada através de reuniões, uma vez que as alterações identificadas

podem necessitar de conhecimentos de vários papéis diferentes.

4.5 IMPLEMENTAÇÃO

A disciplina de Implementação envolve a construção do release, através da geração dos códigos-fonte das aplicações. Ela envolve as atividades mais demoradas e trabalhosas. Além disso, nesta disciplina há um aumento na quantidade de pessoas – geralmente programadores – gerando certa complicação no gerenciamento da equipe. Os artefatos gerados na Disciplina Análise e Projeto constituem a base para realizar as atividades desta disciplina.

Na maioria dos casos, as atividades pertencentes a esta disciplina são as que demandam mais tempo para serem realizadas. A disciplina é constituída pelas seguintes atividades:

- Distribuir Plano de Desenvolvimento para a equipe e sanar possíveis dúvidas;
- Realizar alterações no banco de dados, como criação de novas tabelas, alteração em tabelas já existentes, etc;
- Implementar código-fonte. Um item importante nesta atividade é a necessidade do estabelecimento de padrões para a implementação. Levando-se em conta que cada Programador tem uma personalidade própria e, portanto, uma maneira própria de programar, o não estabelecimento de um padrão pode acarretar na construção de um sistema codificado de maneira totalmente heterogênea. Uma possível manutenção futura se transformará numa tarefa árdua e demorada. Além disso, o estabelecimento de uma padronização garante que todos conseguirão fazer análises de qualquer aplicação, mesmo que não seja o responsável pela codificação da mesma. Esta atividade requer total atenção do Programador e por isso, o ambiente físico no qual se encontra a Equipe de Desenvolvimento deve ser o mais bem preparado possível;
- Revisar o código. Esta atividade tem por objetivo identificar inconsistências e erros na codificação. Além disso, podem ser sugeridas melhorias nos códigos, por exemplo, deixando a aplicação mais leve.

4.6 GERENCIAMENTO DE DOCUMENTAÇÃO PARA OS CLIENTES

Esta disciplina é análoga à Configuração e Controle de Mudanças do RUP. Ela foi alterada para lidar com a documentação para os clientes. Assim, manuais do usuário e changelogs fazem parte de seu escopo de estudo. A disciplina no RUP gerenciava o controle de versões do projeto, mas este item foi retirado da mesma – neste modelo – uma vez que o modelo proposto já é focado no desenvolvimento de versões de um sistema, ou seja, o controle das mesmas pode ser considerado "automático". A disciplina é constituída pelas seguintes atividades:

- Elencar todas as alterações, novos recursos e correções de erros do release;
- Elaborar Carta de Atualização;
- Revisar Carta de Atualização;
- Traduzir Carta de Atualização, quando necessário.

4.7 TESTE

A qualidade, embora tenha se tornado uma característica fundamental dos sistemas, continua sendo um diferencial importante, razão pela qual muitas empresas tem buscado certificações de qualidade. Aliado a isto, recursos complexos surgem a cada dia, fazendo com que o desenvolvimento de software se torne uma tarefa cada vez mais complexa. Quando o nível de complexidade dos sistemas aumenta, a possibilidade do surgimento de erros também aumenta. Por isso, antes que um sistema seja entregue ao cliente, ele deve ser testado ante todas as frentes possíveis de erros e inconsistências.

A disciplina de Teste tem o objetivo de propiciar ambientes úteis de teste para a verificação e análise de todos os itens de uma aplicação. A disciplina é constituída pelas seguintes atividades:

- Elaborar Casos e Procedimentos de Teste;
- Executar Casos e Procedimentos de Teste. Os testes devem ser realizados durante e após a implementação do código-fonte. Quando houver um protótipo os testes já podem ser iniciados;
- Registrar resultados dos testes;
- Avaliar resultados dos testes. Quando algum teste identificar um problema a ser corrigido, a solução deve ser implementada e, depois, os testes devem ser refeitos, constituindo um ciclo.

4.8 AMBIENTE

Grandes projetos possuem o que é chamado de projeto piloto, que consiste na aplicação de um novo projeto em um ambiente para estudo e análise. Ele é importante porque na maioria das vezes o ambiente de desenvolvimento

não possui todos os recursos e limitadores que o ambiente de produção terá.

Esta disciplina tem como objetivo avaliar o release finalizado ante um ambiente real de produção. Embora tenha o mesmo nome de uma disciplina do RUP, seu conceito é completamente diferente. De certa maneira, ela pode ser vista como o último e mais importante teste. Também é importante salientar que as atividades desta disciplina são específicas para cada empresa. A disciplina possui as seguintes atividades:

- Empacotar aplicações. Esta atividade consiste em agrupar todos os módulos do release em um mesmo local;
- Criptografar aplicações, através de um software específico para este fim. A criptografia protege a empresa contra comercialização não autorizada do sistema;
- Finalizar todos os arquivos do release;
- Atualizar servidor piloto, de maneira análoga à que será feita nos clientes;
- Testar servidor piloto. É importante lembrar que qualquer problema identificado nesta fase é complexo, uma vez que todas as aplicações já foram testadas e finalizadas. Erros identificados aqui, geralmente comprometem muito o prazo estimado anteriormente.

5 CONCLUSÃO

As Micro, Pequenas e Médias empresas somam a maior parte entre todas as empresas que desenvolvem software no país. Desta forma é importante identificar as falhas e propor melhorias para o processo de software visto que a qualidade de um software está diretamente ligada ao seu processo de desenvolvimento.

A formalização e documentação de um processo é o primeiro passo na implantação de um modelo de melhoria de software, seja o CMMI (Capability Maturity Model - Integration ou Modelo de Maturidade em Capacitação – Integração) ou o MPS.BR (Melhoria de Processos do Software Brasileiro). O modelo proposto pode ajudar a empresa na implantação de um destes modelos.

Além disso, a formalização de um modelo permite que sejam identificados gargalos no processo, etapas que podem ser realizados em paralelo – diminuindo, assim, o tempo de desenvolvimento –, requisitos necessários para cada um que realizará cada atividade e, principalmente, a identificação mais fácil de erros que poderão ser corrigidos sem comprometer o planejamento realizado.

Ainda, formalizar um processo permite que todas as etapas do desenvolvimento sejam devidamente documentadas. Isto colabora facilitando as atividades de manutenção e adaptação do software bem como a integração de novos membros à equipe.

Este trabalho destacou os principais problemas enfrentados e tais informações podem ser utilizadas para definir um processo de software compatível com as características da empresa analisada.

REFERÊNCIAS

HILGERT, F.P.; MOREIRA, L.S.R.; PRIKLADNICKI, R.; BOSSLE, R.; MÓRA, M.C.; BACK, R. Gestão de Mudança em Melhoria de Processo de Software: Um Relato de Integração entre RH e SEPG na Tlantic SI. In: Workshop Um Olhar Sociotécnico sobre a Engenharia de Software, IV, 2008. Florianópolis, 2008.

NASCIMENTO, H.A.A. Melhoria do Processo de Software e a Avaliação da Maturidade no Modelo MPS.BR em uma Pequena Empresa. 2008. 61f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2008.

SILVA, E.L.; MENEZES, E.M. Metodologia da Pesquisa e Elaboração de Dissertação. Florianópolis, 2005. 139f.