# Plano de Colaboração

*Tópicos Avançados em Engenharia de Software, CIn - UFPE, 2013.2*

*Novembro de 2013*

Cleydyr Bezerra de Albuquerque (cba)
João Guilherme Farias Duda (jgfd)



## 1. Sobre este Documento

O plano de colaboração explana atividades a ser realizadas pela equipe relacionadas à colaboração com o projeto Research Group Management System (RGMS) [1], no âmbito da disciplina de Tópicos Avançados em Engenharia de Software [2]. O processo de colaboração utilizará ferramentas auxiliares, de acordo com as diretizes estabelecidas para a disciplina, como Git [3] e GitHub [4].

Este documento está organizado como segue: A seção 2 apresenta práticas que serão seguidas durante todas as etapas. A seção 3 apresenta detalhes de cada etapa e o cronograma a ser cumprido. A última seção é constituida de *links* que podem ser úteis durante o processo.

## 2. Guias Gerais de Desenvolvimento

Os principais aspectos do desenvolvimento serão estabelecidos na próximas sub-seções; a saber: comunicação, divisão de responsabilidades e tempo de dedicação.

### 2.1 Comunicação

A comunicação se dará principalmente através de e-mail e serviços de mensagem instantâneas. A partir da etapa de correção falhas (seção 3.2), acontecerão, semanalmente, seções de *pair programming* – presenciais ou através de videoconferência e *screen sharing*. Datas e horários dos encontros serão fixados a cada de duas semanas, de acordo com a demanda da etapa em questão e disponibilidade dos integrantes.

### 2.2 Divisão de Responsabilidades

Os membros da equipe compartilham das mesmas responsabilidades quanto à execução das atividades estabelecidas na seção 3. Além da execução dessas, firma-se o compromisso de atender às exigências do projeto quanto aos padrões de:

1. colaboração através do GitHub;
1. desenvolvimento de código;
2. documentação do sistema;
3. arquitetura e design estabelecidos;

É importante salientar que, devido ao escopo limitado da documentação atual do sistema [5][6], parte substancial do esforço de colaboração será aplicada à possível elaboração e documentação de padrões (seção 3.1).

### 2.3 Local e Horas de Trabalho

Cada membro da equipe se propõe a dedicar 6 horas semanais em regime de *home-office* para a execução das atividades citadas na seção 3. Adicionalmente, haverá o tempo dedicado a atividades realizadas em conjunto, que ocorrerão no Centro de Informática.

## 3. Cronograma de Atividades

A realização das atividades de colaboração está dividida em cinco etapas, de acordo com a estrutura seguinte. Atenção especial deve ser dada às possíveis depedências entre as atividades.

| Identificador do Grupo de Atividades | Data de Início | Data de Término |
|---|---|---|
| 3.1 | 27/11/2013 | 11/12/2013 |
| 3.2 | 09/11/2013 | 13/01/2014 |
| 3.3 | 20/01/2013 | 29/01/2014 |
| 3.4 | 05/02/2014 | 12/02/2014 |
| 3.5 | 17/02/2014 | 26/02/2013 |

### 3.1 Estudo e Documentação da Arquitetura do Sistema

A primeira etapa de colaboração abrange o estudo do código (familizarização com bibliotecas, padrões e arquitetura utilizados) e documentação. Para o estudo, pretende-se utilizar as mesmas ferramentas utilizadas para o desenvolvimento (IDEs com suporte ao Grails, terminal de linha de comando e editores de texto).

O sistema (arquitetura, guias de colaboração, desenvolvimento, implantação, testes, etc.) deve ser documentado principalmente através da Wiki do projeto, no Github [7] -- onde poderá ser consultado livremente e facilmente construído de maneira colaborativa.

### 3.2 Identificação e Correção de Falhas em Funcionalidades Existentes no Sistema

Esta etapa abrange atividades para identificação de falhas no sistema e criação de testes que as exteriorizem. Vale notar possívei interseções entre a etapa anterior (de estudo da arquitetura) e esta, pois estas falhas podem ser detectadas durante atividades de estudo e devidamente indicadas na ferramenta de *issues* do Github [8].
Além da identificação e criação de testes, esta etapa também envolve a resolução de problemas encontrados pela equipe ou outros problemas que registrados encontrados por outros colaboradores.

### 3.3 Identificação e Correção de Problemas de Reuso no Código do Sistema

A identificação de problemas de reuso no código será realizada através de ferramentas para detecção automatizada de clones (e.g.: Simian [9], CCFinder [10] e a IDE utilizada no desenvolvimento, IntelliJ [11]) e de análise estática (a saber, CodeNarc [12] e IntelliJ [11]).
Os resultados das análises de código devem ser registrados, bem como problemas detectados de forma não automática.
Por fim, devem ser selecionados problemas sujeitos a refatoração, que deve usar padrões de projeto adequados e implementar testes sempre que viável [13]. É importante registrar código anterior e posterior a refatoração, padrões de projeto utilizados (com justificativa para a utilização) e quaisquer diretrizes necessárias para a utilização do código modificado.

### 3.4 Identificação e Correção de Problemas de Modularidade no Código do Sistema

A identificação de problemas de modularidade será realizada através da utilização de ferramentas para coleta de métricas de código [14][15]. Problemas apontados pelas ferramentas ou detectados manualmente pela equipe devem ser registrados.
Posteriormente, alguns desses problemas são selecionados para refatoração, utilizando os mesmos padrões de refatoração explanados na seção 3.3.

### 3.5 Identificação e Implementação de Novas Funcionalidades e Melhorias

Esta etapa consiste na especificação e implementação de novas *features d*o projeto. É importante levar em consideração o modelo de *features* do RGMS e a utilização do pré-processador Velocity [16] para compilação condicional.
Os membros da equipe devem estar atentos à implementação de novas funcionalidades para que ela não altere o comportamento de outras funcionalidades ou introduza problemas de reuso e modularidade.

## 4. Referências


[1] RGMS no Github https://github.com/spgroup/rgms

[2] Tópicos Avançados em Engenharia de Software
https://sites.google.com/a/cin.ufpe.br/desenvolvimento-de-software-taes-2013-2/

[3] Git http://git-scm.com/

[4] Github https://github.com/

[5] RGMS Development Process https://docs.google.com/a/cin.ufpe.br/viewer?a=v&pid=sites&srcid=Y2luLnVmcGUuYnJ8bGluaGFzLWRlLXByb2R1dG9zLWRlLXNvZnR3YXJlLXRhZXMtMy0yMDEyLTF8Z3g6NGYxYTk1ZTdjOWI0YzQ2OQ

[6] RGMS Development Process Specification https://docs.google.com/a/cin.ufpe.br/viewer?a=v&pid=sites&srcid=Y2luLnVmcGUuYnJ8bGluaGFzLWRlLXByb2R1dG9zLWRlLXNvZnR3YXJlLXRhZXMtMy0yMDEyLTF8Z3g6NjUxY2E2YTRhOWJlZGViOA

[7] Wiki do RGMS no Github https://github.com/spgroup/rgms/wiki

[8] Issues do RGMS no Github https://github.com/spgroup/rgms/issues

[9] Simian -- Similarity Analyser http://www.harukizaemon.com/simian/

[10] CCFinder http://www.ccfinder.net/ccfinderxos.html

[11] IntelliJ IDEA http://www.jetbrains.com/idea/

[12] CodeNarc http://grails.org/plugin/codenarc

[13] Spock https://code.google.com/p/spock/

[14] GMetrics http://www.grails.org/plugin/gmetrics

[15] SourceMiner https://wiki.dcc.ufba.br/pub/LES/GlaucoCarneiro/org.gesa.sourceminer_2.17.0.jar

[16] Velocity http://velocity.apache.org/