

Universidade Federal Fluminense

Instituto de Computação

Coordenação do Curso de Pós-Graduação em Computação

**Proposta de Dissertação de Mestrado**

Título

DyeVC: Em busca de métricas que determinem a ordem ideal de junção em Sistemas de Controle de Versão Distribuídos

Aluno

Cristiano Machado Cesário

Orientador

Leonardo Murta

# Descrição do Tema

Os Sistemas de Controle de Versão (SCV) remontam desde a década de 70, quando surgiu o SCCS [Rochkind 1975] e têm como principal objetivo permitir o desenvolvimento controlado de software por diversos desenvolvedores, mantendo todo o seu histórico [Estublier 2000]. Nestes quase 40 anos, os SCVs evoluíram de uma abordagem com repositório centralizado e acesso local (como era o acesso ao sistema SCCS e ao sistema RCS [Tichy 1985]) para uma abordagem centralizada cliente-servidor (como é o caso do CVS [Cederqvist 2003] e do Subversion [Collins-Sussman et al. 2008]) e, mais recentemente, para uma abordagem distribuída (SCV distribuídos – SCVD), em que o repositório pode estar espelhado em diversos locais (como é o caso dos sistemas Git [Chacon 2009] e Mercurial [O’Sullivan 2009]).

Segundo [Eclipse Foundation 2011], apesar do Subversion ainda ser o sistema dominante na comunidade *open source*, sua utilização caiu de 58,3% em 2010 para 51,3% em 2011 (uma queda de 12%). Nesse mesmo período, Git e Mercurial (principais SCVDs em uso atualmente) aumentaram sua participação de 9,8% para 17,4% (um aumento de 77,5%). Esses dados demonstram uma tendência cada vez maior de utilização de SCVDs em comunidades *open source*.

A criação de ramos, que já era necessária para o desenvolvimento de softwares complexos em SCVCs [Walrad e Strom 2002] torna-se um padrão recorrente ao utilizar SCVDs, pois cada clone do repositório pode ser visualizado como um novo ramo criado implicitamente, no qual os desenvolvedores podem ainda criar explicitamente novos ramos. Os ramos também podem ser criados no momento em que são trazidas atualizações do repositório de origem para o repositório local (operação conhecida como *pull*), ou quando as alterações realizadas no repositório local são enviadas para outro repositório (operação conhecida como *push*). Estes ramos, em algum momento, poderão ser reintegrados à sua origem (efetuando operações de junção, ou *merge*), para refletir as alterações realizadas.

A proliferação de ramos, o crescente aumento das equipes de desenvolvimento e sua crescente dispersão em locais distantes – até mesmo em diferentes continentes –, tornam árdua a tarefa de se administrar a evolução de um sistema complexo, na medida em que dificultam a identificação:

* Dos clones que foram criados a partir de um repositório;
* Das dependências entre os diferentes clones;
* Da ordem de junção entre os diferentes clones / ramos que resulte no menor esforço;
* Das alterações que estão sendo feitas em paralelo (em diferentes clones ou em diferentes ramos); e
* Dos conflitos físicos, sintáticos e semânticos que podem ser causados ao efetuar a junção das alterações realizadas concorrentemente.

# Resultados Esperados

A proposta desta Dissertação de Mestrado consiste em uma abordagem chamada DyeVC (*Dye over Version Control*), uma analogia à utilização de corantes (do inglês *dye*) em células para observar o processo de divisão celular. O objetivo da abordagem DyeVC é visualizar como uma rede de clones de SCVDs evolui e apresentar métricas que identifiquem o esforço de junção e a ordem ideal de junção entre diferentes ramos, proporcionando aos usuários uma forma de controlar essa evolução através da identificação de alterações concorrentes em diferentes clones, da percepção proativa de conflitos e do relacionamento emergente entre os diferentes clones de um determinado sistema.

## Evolução da rede

A abordagem DyeVC pretende apresentar as informações da rede em três diferentes níveis de detalhe:

* **Nível 1:** Visão topológica, na qual será possível identificar os repositórios existentes e como as atualizações são enviadas / recebidas entre eles, como mostrado na Figura 1, que apresenta um cenário com um conjunto de desenvolvedores, cada um possuindo um clone de um repositório que foi criado originalmente no Instituto Xavier. As setas na Figura 1 indicam o sentido no qual as atualizações são enviadas. Assim, por exemplo, Vampira pode solicitar atualizações de Gambit (*pull*)e, por sua vez, suas atualizações podem ser enviadas para Gambit e Fera (*push*);
* **Nível 2:** Situação de um repositório local em relação a um repositório remoto, baseada na existência de *commits* não replicados em cada um deles; e
* **Nível 3:** Situação de um ramo específico, em relação à cópia de trabalho do desenvolvedor e em relação aos seus correspondentes remotos (caso existam).

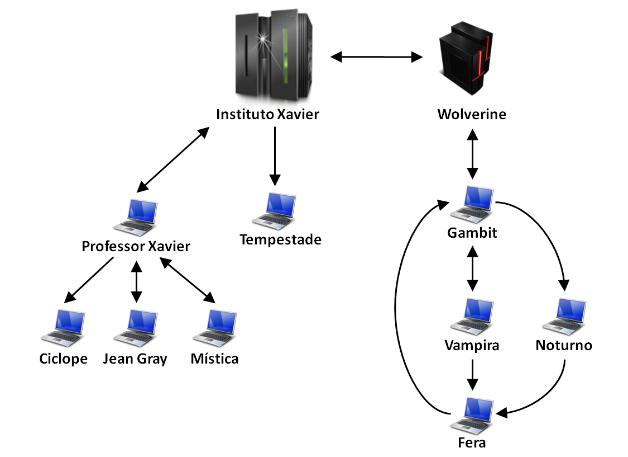


Figura – Uma visão topológica de clones de um repositório distribuído

Esses níveis foram identificados durante o Estudo Orientado, no qual também foram identificadas e implementadas as verificações de possíveis situações de um repositório (Nível 2). As situações identificadas estão representadas na Tabela 1.

Tabela – Situações possíveis de um repositório

| **Situação** | **Descrição** |
| --- | --- |
| nocheck_32 | Repositório inválido. Isso ocorre quando o DyeVC não consegue acessar o projeto no local definido pelo usuário. |
| question_32 | Repositório ainda não avaliado pelo DyeVC. |
| check_32 | O repositório encontra-se sincronizado com todos aqueles com os quais se comunica. |
| ahead_ylw_32 | O repositório possui modificações que não foram enviadas aos clones de origem. |
| behind_ylw_32 | O repositório encontra-se defasado em relação aos clones de origem, ou seja, há modificações que não foram trazidas. |
| aheadbehind_ylw_32 | O repositório possui modificações que não foram enviadas aos clones de origem, mas também está defasado, necessitando que modificações que ocorreram remotamente sejam trazidas. |

Para obter os dados necessários para apresentar as informações do Nível 1, esse trabalho propõe que existam agentes DyeVC nas máquinas de cada desenvolvedor, os quais enviariam esses dados a um banco de dados central (Figura 2), onde a visão local de cada desenvolvedor seria consolidada em uma visão global da topologia.

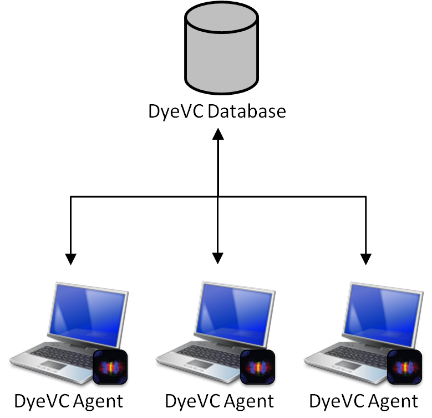


Figura – Arquitetura conceitual da abordagem DyeVC

## Métricas para cálculo do esforço de junção

Segundo [Santos 2012], o esforço de junção entre dois ramos possui uma correlação direta com a quantidade de artefatos modificados em comum e com a quantidade de conflitos físicos existentes entre esses ramos. À medida que a história de dois ramos divergentes se torna mais longa, isto é, à medida que existe uma quantidade maior de *commits* em cada ramo antes da junção, é esperado que a quantidade de conflitos físicos também aumente.

Essa proposta, portanto, utiliza a quantidade de conflitos físicos como base para cálculo do esforço de junção, acrescentando, ainda, a **quantidade de *commits* não sincronizados entre ramos** como um indicador da possibilidade de ocorrência de conflitos físicos (quanto maior o valor dessa métrica, maior a probabilidade de ocorrência de conflitos físicos e, consequentemente, maior o esforço de junção).

## Mecanismo de análise

A Figura 3 apresenta a proposta de mecanismo de análise da ferramenta DyeVC. Inicialmente deve ser criada uma replicação (clone) do repositório que se deseja analisar. Este clone será utilizado como repositório de trabalho da ferramenta. Periodicamente, a ferramenta irá baixar os novos *commits* dos repositórios remotos e as atualizações de topologia do banco de dados central. Em seguida, para cada ramo existente no repositório de trabalho, será feita a análise para identificar a melhor ordem de junção. Ao terminar a análise, o banco de dados central é atualizado.

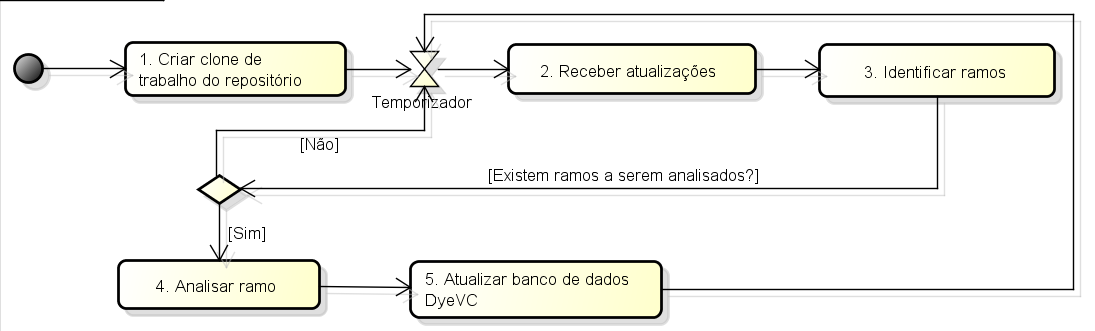


Figura – Passos da abordagem DyeVC

# Implementação

A implementação do protótipo da abordagem DyeVC desenvolvido no Estudo Orientado utiliza a tecnologia Java Web Start[[1]](#footnote-1) e é capaz de monitorar repositórios distribuídos criados com o Git*.* A obtenção de informações dos repositórios é realizada utilizando-se a biblioteca JGit[[2]](#footnote-2). A aplicação é executada de maneira não intrusiva, na área de notificação do sistema operacional (Figura 4). Em relação à codificação, utiliza-se a linguagem Java 1.6.

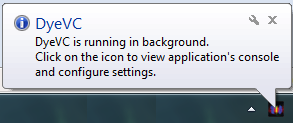


Figura 4 – DyeVC executado na área de notificações

# Cronograma de Atividades

**Primeiro semestre/2013:**

| **Atividades** | **Mar** | **Abr** | **Mai** | **Jun** | **Jul** |
| --- | --- | --- | --- | --- | --- |
| Implementação de visualização do log de repositório |  |  |  |  |  |
| Implementação de métrica de distância entre commits |  |  |  |  |  |
| Elaboração de proposta de dissertação de mestrado |  |  |  |  |  |
| Implementação de layout customizado para visualização do log |  |  |  |  |  |
| Implementação de filtros para colapsar o log |  |  |  |  |  |
| Pesquisa de mecanismo de descoberta da topologia de um projeto |  |  |  |  |  |
| Pesquisa de outras métricas construção de algoritmo de ordem de junção (métricas) |  |  |  |  |  |
| Implementação de mecanismo de descoberta da topologia |  |  |  |  |  |
| Implementação da GUI para apresentar a topologia |  |  |  |  |  |
| Implementação de métricas para ordem de junção |  |  |  |  |  |
| Elaboração de artigo para workshop |  |  |  |  |  |

**Segundo semestre/2013:**

| **Atividades** | **Ago** | **Set** | **Out** | **Nov** | **Dez** |
| --- | --- | --- | --- | --- | --- |
| Executar experimentos utilizando projetos *open source* |  |  |  |  |  |
| Análise dos resultados dos experimentos |  |  |  |  |  |
| Elaboração de artigo com resultados dos experimentos |  |  |  |  |  |
| Elaboração da dissertação de mestrado |  |  |  |  |  |
| Revisão da dissertação de mestrado |  |  |  |  |  |
| Apresentação da dissertação de mestrado |  |  |  |  |  |

# Referências

Cederqvist, P. (2003). Version Management with CVS. Free Software Foundation.

Chacon, S. (2009). *Pro Git*. 1. ed. Berkeley, CA, USA: Apress.

Collins-Sussman, B., Fitzpatrick, B. W. e Pilato, C. M. (2008). *Version Control with Subversion*. Sebastpol, CA, USA: O’Reilly Media. v. 2

Eclipse Foundation (Jun 2011). The *Open source* Developer Report - 2011 Eclipse Community Survey.

Estublier, J. (2000). Software configuration management: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE ’00. ACM.

O’Sullivan, B. (2009). *Mercurial: The Definitive Guide*. 1. ed. O’Reilly Media.

Rochkind, M. J. (1975). The Source Code Control System. *IEEE Transactions on Software Engineering (TSE)*, v. 1, n. 4, p. 364–370.

Santos, R. de S. (2012). Avaliação do Esforço de Junção de Ramos em Sistemas de Controle de Versão. Universidade Federal Fluminense - UFF.

Tichy, W. (1985). RCS: a system for version control. *Software - Practice and Experience*, v. 15, n. 7, p. 637–654.

Walrad, C. e Strom, D. (set 2002). The importance of branching models in SCM. *Computer*, v. 35, n. 9, p. 31 – 38.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.: Leonardo Murta

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Aluno: Cristiano Machado Cesário

1. http://docs.oracle.com/javase/6/docs/technotes/guides/javaws/ [↑](#footnote-ref-1)
2. http://www.eclipse.org/jgit/ [↑](#footnote-ref-2)