PROPOSTA DE CRIAÇÃO DE UMA BASE ESTRUTURADA PARA ACOMPANHAMENTO DE PROJETO - *ENGINEERING TOOLS*

Rafael Del Col Carlet [1]; Vanderlei Cunha Parro [2]

[1] Aluno de Iniciação Científica da Escola de Engenharia Mauá (EEM/CEUN-IMT);
[2] Professor da Escola de Engenharia Mauá (EEM/CEUN-IMT).

Resumo. *Este projeto de iniciação científica (IC) está ligado a projeto de pesquisa financiado pelo CNPq sob a coordenação do INCT - INESPAÇO. O projeto proposto ao CNPq com o título:* ***Desenvolvimento de um modelo de laboratório de uma plataforma multifuncional para avaliação de algoritmos de processamento de sinais e protocolos de comunicação para aplicações espaciais.*** *Visa a construção de um modelo de laboratório de uma plataforma para testes de algoritmos para serem embarcados. Este projeto de IC tem o objetivo a estruturação de uma base de software de apoio, fundamentada em software livre, que possibilite a implementação adequada da metodologia Spice for Space [1].*

Introdução

Muitos projetos de engenharia são desenvolvidos sem estarem otimizados, tanto em questão de tempo, quanto de custos. Um dos fatores que influência esta característica negativa é porque não houve uma análise das ferramentas de suporte a serem utilizadas durante a realização dos projetos. Esta iniciação consiste em estruturar uma proposta que evite este aspectos negativos, sendo seu estudo focado na viabilização da utilização destas ferramentas em projetos, sobretudo aqueles que estão relacionados com estudos espaciais. Para realizar um estudo mais aprofundado foi necessário limitar o escopo de ferramentas e será detalhado abaixo. Mesmo com esta limitação, é possível que haja uma ampliação de escopo na medida em que os estudos das ferramentas atuais encerrem. O sistema desenvolvido pela European Space Agency (ESA) e conhecido como *spice for space* introduz de forma estruturada este conceito. As categorias das ferramentas de controle foram baseadas nas do projeto CoRoT.

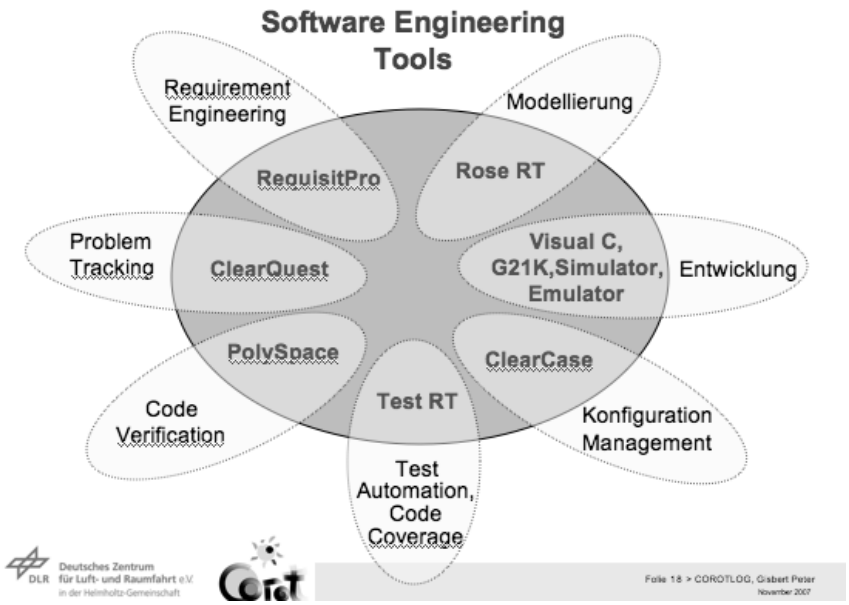


Figura 1 – Programas utilizados no CoRoT - Fonte: http://www.lesia.obspm.fr/perso/claude-catala/plato_web.html

Materiais e Métodos

Para a determinação de quais ferramentas são mais eficientes, considerando a área espacial, foi realizado um intenso estudo de diversos tipos de sistemas. A variedade de classes das ferramentas estudadas foi ampla e incluiu o gerenciamento de: Requisitos, Versões, Projetos, Erros *Runtime e* Modelagem de Software. Em suma, instalou-se tais sistemas em um ambiente de teste para que suas características e vantagens pudessem ser simuladas e comparadas na prática. Uma característica importante do projeto de pesquisa foi a preferência por ferramentas de código aberto, visando um aumento ainda maior na eficiência do projeto do ponto de vista de independência. Este aumento foi buscado focando em sistemas que ampliassem a integração entre todos os participantes do projeto e a onisciência entre as tarefas deles.

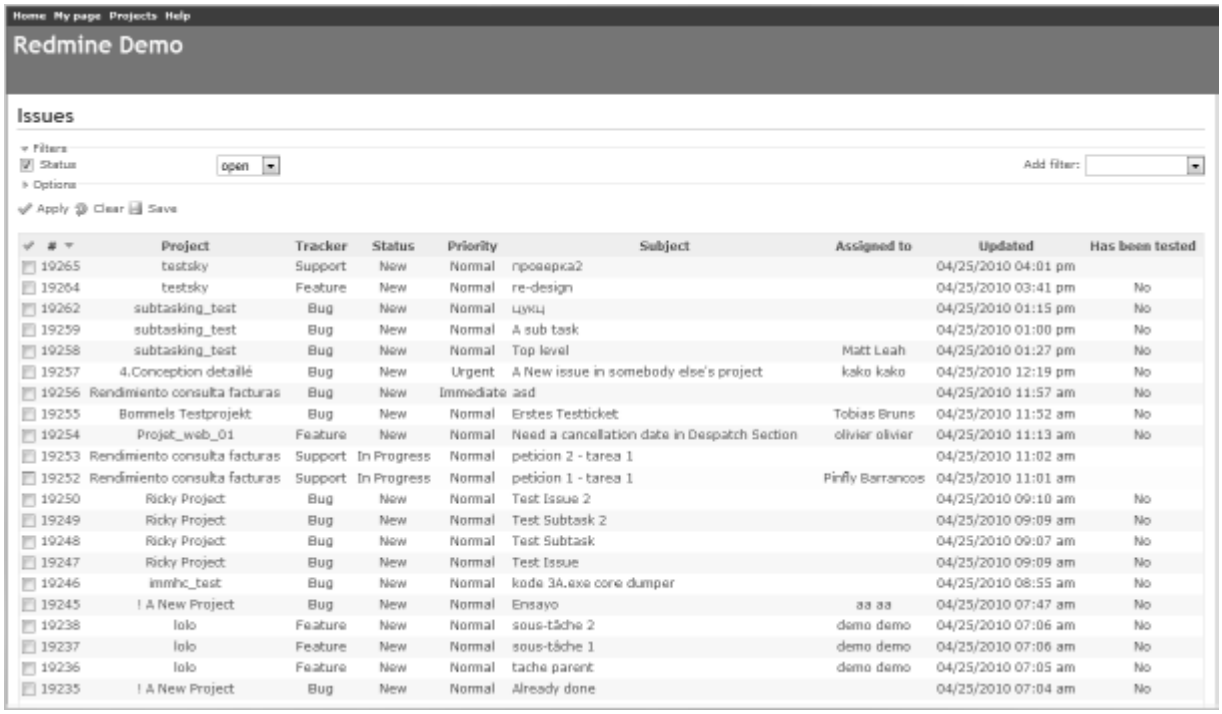
Introdução

A prioridade foi concedida aos sistemas de controle de versões e de requisitos de *software*. Tais classes de ferramentas tiveram preferência por possuírem uma maior importância e aplicabilidade do que os outros tipos, uma vez que estes *softwares* podem ser utilizados em uma gama maior de projetos.

As características gerais dos sistemas de controle de versões são a presença de histórico de versões do arquivo, a facilidade de retorno a versões estáveis e a economia de tempo, sendo este realizado principalmente pelo fato de diversas pessoas poderem trabalhar sobre os mesmos documentos simultaneamente. Foi dada prioridade para *softwares open source* mesmo estando ciente da vantagem dos *softwares* comerciais, que consiste na responsabilidade sobre a perda dos dados. Dentro deste sistema, foram analisados dois grandes tipos: Centralizado e Distribuído. As principais características do sistema centralizado consistem na presença de um servidor que armazena todo o histórico do trabalho e os clientes, que possuem apenas a cópia da última versão do código. Quando o programador desejar, ele pode realizar um comando de *commit* que gera uma nova versão no servidor e arquiva a anterior. Em oposição à este sistema existe o escopo Distribuído, estrutura esta que armazena

todas as versões dos arquivos no computador do programador. As vantagens e desvantagens deste tipo serão melhores desenvolvidas no item "Controle de Versões". Um conceito interessante desenvolvido durante o estudo consistiu na percepção de que este tipo de *software* de controle pode ser utilizado para qualquer tipo de arquivos, dando-se preferência para arquivos de textos.

Outro sistema de software muito estudado foi a classe de requisitos de software. Estes sistemas têm extrema importância na realização de projetos, uma vez que diminuem o tempo, os riscos e oferecem uma melhor visualização do projeto. Com isso, é possível alcançar um nível mais alto de organização e onisciência entre toda a equipe, o que aumenta a integração, ponto crucial na programação conjunta. Este sistema possibilita a organização de tarefas, incluindo seus detalhes, status, prioridade e responsável pela tarefa, podendo estas tarefas serem organizadas por todos estes critérios descritos. Um exemplo desta organização, obtida por um dos *softwares* estudados, pode ser encontrado na Figura 2.



#	Project	Tracker	Status	Priority	Subject	Assigned to	Updated	Has been tested
19265	testsky	Support	New	Normal	проверка2		04/25/2010 04:01 pm	
19264	testsky	Feature	New	Normal	re-design		04/25/2010 03:41 pm	No
19262	subtasking_test	Bug	New	Normal	цукц		04/25/2010 01:15 pm	No
19259	subtasking_test	Bug	New	Normal	A sub task		04/25/2010 01:00 pm	No
19258	subtasking_test	Bug	New	Normal	Top level	Matt Leah	04/25/2010 01:27 pm	No
19257	4.Conception detaillé	Bug	New	Urgent	A New issue in somebody else's project	kako kako	04/25/2010 12:19 pm	No
19256	Rendimiento consulta facturas	Bug	New	Immediate	asd		04/25/2010 11:57 am	No
19255	Bommels Testprojekt	Bug	New	Normal	Erstes Testticket	Tobias Bruns	04/25/2010 11:52 am	No
19254	Projet_web_01	Feature	New	Normal	Need a cancellation date in Despatch Section	olivier olivier	04/25/2010 11:13 am	No
19253	Rendimiento consulta facturas	Support	In Progress	Normal	peticion 2 - tarea 1		04/25/2010 11:02 am	
19252	Rendimiento consulta facturas	Support	In Progress	Normal	peticion 1 - tarea 1	Pinfly Barrancos	04/25/2010 11:01 am	
19250	Ricky Project	Bug	New	Normal	Test Issue 2		04/25/2010 09:10 am	No
19249	Ricky Project	Bug	New	Normal	Test Subtask 2		04/25/2010 09:09 am	No
19248	Ricky Project	Bug	New	Normal	Test Subtask		04/25/2010 09:07 am	No
19247	Ricky Project	Bug	New	Normal	Test Issue		04/25/2010 09:09 am	No
19246	immhc_test	Bug	New	Normal	kode 3A.exe core dumper		04/25/2010 08:55 am	No
19245	! A New Project	Bug	New	Normal	Ensayo	aa aa	04/25/2010 07:47 am	No
19238	lolo	Feature	New	Normal	sous-tâche 2	demo demo	04/25/2010 07:06 am	No
19237	lolo	Feature	New	Normal	sous-tâche 1	demo demo	04/25/2010 07:06 am	No
19236	lolo	Feature	New	Normal	tache parent	demo demo	04/25/2010 07:05 am	No
19235	! A New Project	Bug	New	Normal	Already done		04/25/2010 07:04 am	No

Figura 2 – Aplicativo Redmine listando requisitos - Fonte: Infraestrutura Local

Controle de Versões

Foram analisados os programas: Concurrent Version System (CVS), Subversion (SVN), Git, Bazaar, Mercurial e MediaWiki. O CVS constituiu a primeira era de *softwares* de gerenciamento de versões com funcionalidades consideráveis do ponto de vista produtivo. Constitui um sistema tradicional baseado no *Revision Control System* que geralmente tem o seu servidor rodando em plataforma Unix, já possuía um log da manipulação dos arquivos, entretanto possuía o empecilho da dificuldade de manipulação com arquivos binários/*Unicode.* O Subversion[2] é uma versão muito mais desenvolvida e pode ser considerado o substituto moderno do CVS uma vez que ele possui uma comunidade muito mais ativa, aumentando assim o suporte. Com isso, há uma ampla gama de ambientes *Guided User Interface*, o que popularizou consideravelmente este sistema. Outras características interessantes são a facilidade da criação de um servidor web para hospedar o histórico de versões e a integração com diversas *IDE – Integrated Development Environment*, incluindo o poderoso Eclipse.

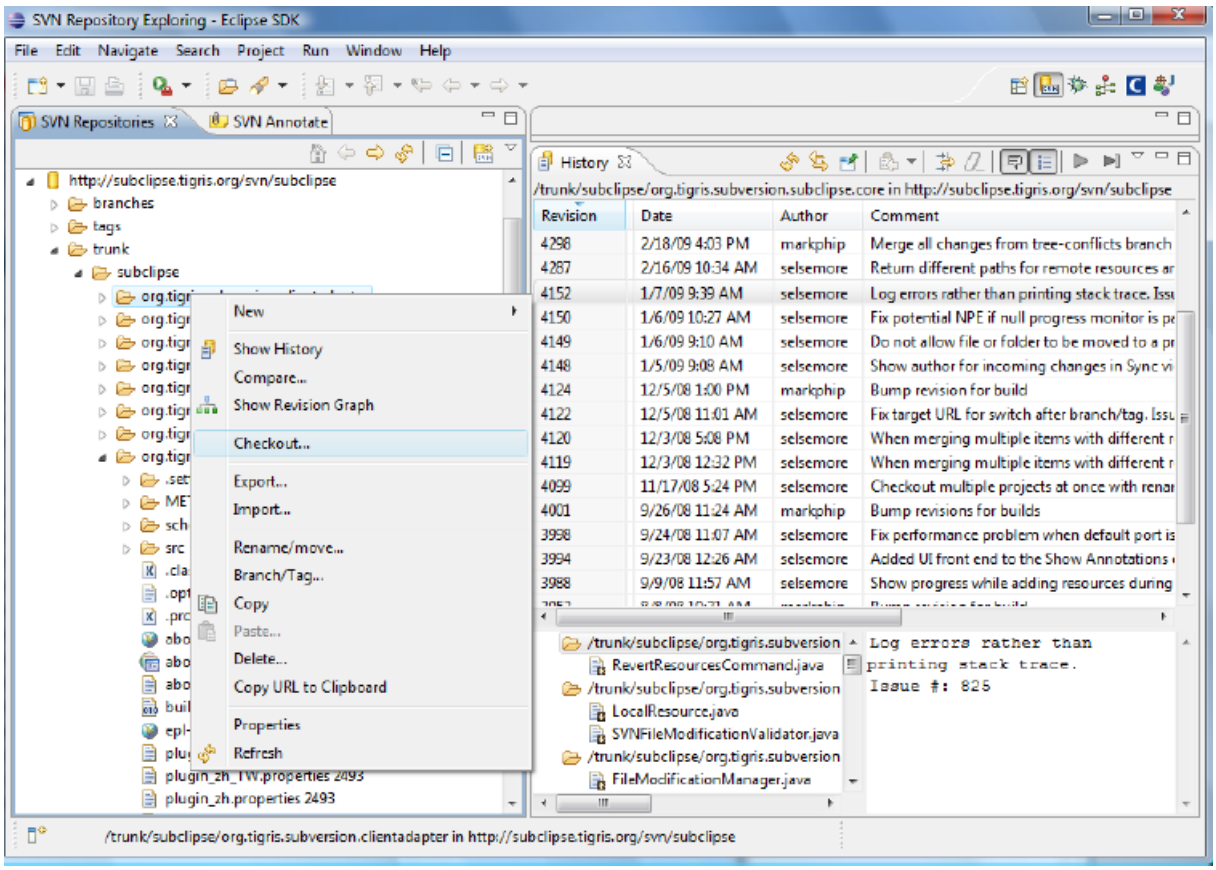


Figura 3 – Subversion implementado na IDE Eclipse. Fonte: http://subclipse.tigris.org/

Outro sistema de controle de versões analisado foi o Git[3]. Este sistema se opõe ao anterior pelo fato de ser descentralizado, o que faz com que todas as cópias fiquem armazenadas em todas as máquinas dos programadores. Com esta característica aparecem algumas vantagens como, por exemplo, o aumento na velocidade, uma vez que todas as versões estão armazenadas localmente, entretanto há o empecilho do espaço, uma vez que projetos grandes podem gerar uma quantidade gigantesca de dados, o que sobrecarregaria a máquina do programador. Entretanto, o sistema Git possui um sistema de compactação que diminui consideravelmente o tamanho do repositório e está descrito na tabela 1.

Tabela 1 - Tamanhos dos Repositórios Git e SVN

Programa	Tamanho
Git	420 Mb
SVN	12Gb

Tabela 2 - Tempos de Clone Git e SVN

Programa	Tamanho
Git	1m 59s
SVN	1m 4s
Bazaar	5m 11s

Outra alternativa de software que apresenta uma gama interessante de recursos foi o Bazaar. A sua principal característica é o seu suporte nativo ao Windows, ou seja, ele não necessita de nenhuma extensão para ser executado no sistema operacional da Microsoft e, como este era predominante dentre os projetos em que a ferramenta seria aplicada, tal característica teve um peso considerável. Não apenas isto mais é importante comentar também sobre a presença de criptografia e a compatibilidade com os protocolos FTP e SSH.

Dentre estes sistemas foi escolhido o Subversion para ser utilizado nos projetos espaciais, pelas características de estabilidade, uma vez que existe uma ampla comunidade dedicada a criação de extensões para este software, o que torna o seu uso muito mais viável. Alem disto, é importante cogitar sobre as características destas extensões que permitem o gerenciamento dos usuários, comparação em tempo real entre os códigos de duas versões distintas, e a comparação de duas ou mais versões, com a possibilidade de unir todas elas.

Requisitos de projeto

Apesar de ter sido realizado o estudo do software pago Rational RequisitePro, deu-se mais importância para as suas alternativas open source, foco principal desta iniciação cientifica. Entre eles se encontra o clássico TRAC [5], sistema escrito em Python que seria o semelhante ao SVN de sua categoria, uma vez que está em uma fase estável. Este sistema em si é relativamente básico, o que não inviabiliza o fato de ele ser muito poderoso e poder aumentar exponencialmente a produtividade. Além disto, existem os chamados Trac-Hacks [6], modificações realizadas pela gigante comunidade de desenvolvedores do sistema, que permite aumentar ainda mais as funcionalidades do Trac. É importante notar que este sistema é utilizado por grandes empresas que prezam a organização de informação, uma vez que este sistema tem o poder de organizar as informações e atividades que estão sendo realizadas no projeto. Outra função interessante e que pesou bastante na decisão por este sistema foi a existência de uma integração SVN-Trac, o que nos permite visualizar o repositório e o arquivo de revisões direto da interface do Trac, sem contar que é possível anexar uma atividade (ou diversas) à um requisito. Por exemplo, se na versão 1.0 o código possuía um bug, e na versão 1.1 este bug foi corrigido, é possível criar um ticket no Trac que descreva este bug e já mostre a parte do código que foi alterada para que tal falha fosse solucionada. Esta informação tem muita utilidade do âmbito da programação em grupo, uma vez que torna desnecessária a comunicação direta entre os programadores, o que gera uma perda de tempo. O adversário direto deste recurso é o poderosíssimo Redmine[7]. Escrito em Ajax Ruby ele oferece suporte a mais de um projeto simultaneamente, recurso inviável no Trac. Tirando alguns detalhes dependentes de aplicações, o Redmine pode ser considerado um Trac Clone, uma vez que segue a linha de desenvolvimento deste sistema, imitando diversos recursos. Há uma vasta gama de recursos que foram adicionados e aprimorados do Trac como, por exemplo, a notificação por e-mail e o gerenciamento de usuários. Entretanto, por ser um sistema mais novo, não possui tantos adeptos e nem a estabilidade adquirida com o tempo pelo Trac. Por isso, optou-se pelo Trac para a monitoria dos recursos dos projetos espaciais.

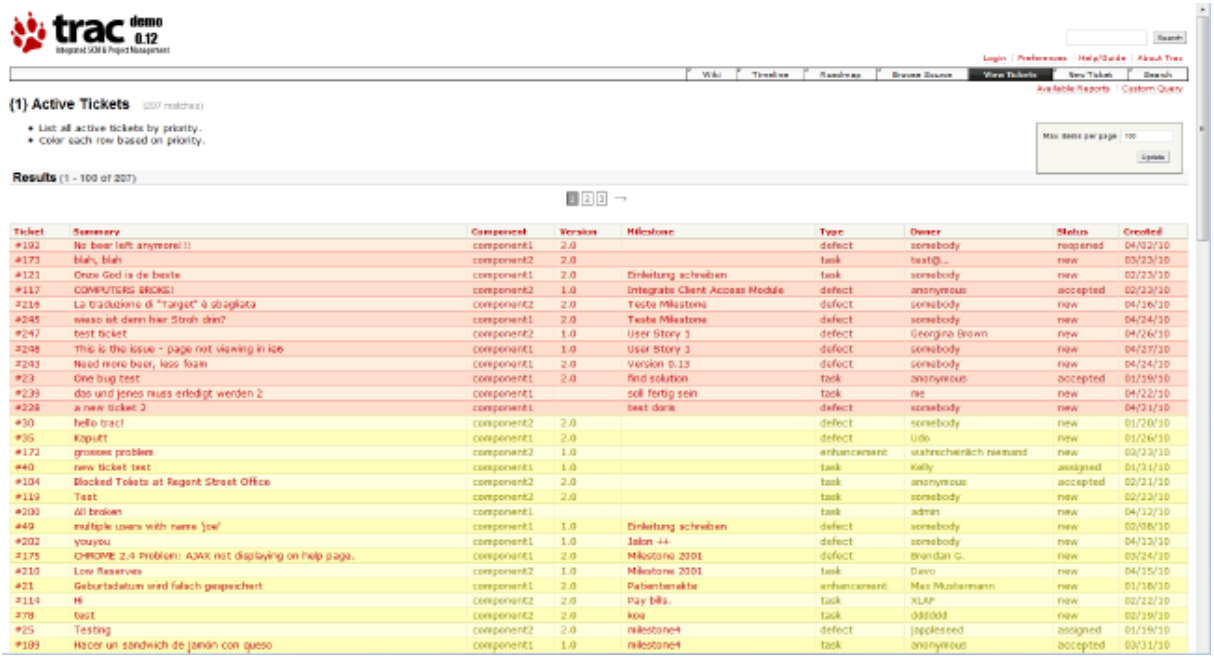


Figura 3 – Sistema escolhido para gerenciamento de requisitos (TRAC) implementado - Fonte: Infraestrutura Local

Pontos em aberto

Os sistemas para controle de Projetos, Erros *Runtime e* Modelagem de Software, como descritos na introdução tiveram a sua decisão prorrogada, uma vez que espera-se um estudo profundo das ferramentas que tiveram sua implementação decidida. Após a conclusão dos testes de viabilidade e produtividade os sistemas de requerimentos e controle de versões, deve-se iniciar uma nova etapa aonde as outras classes de sistemas serão analisadas e devidamente implementadas.

Resultado e Discussão

Por fim, com bases comparativas de todos os sistemas *open source* analisados, instalou-se uma ferramenta de cada categoria nos ambientes de programação e planejamento de outros projetos de pesquisas do Instituto Mauá de Tecnologia e a mudança constatada pela equipe consistiu em um grande avanço nos âmbitos de organização e eficiência dos projetos. A mudança foi grandiosa se considerarmos o aumento na velocidade de evolução do projeto, visto que as ferramentas instaladas aumentaram a organização e integração entre os participantes do estudo, características imprescindíveis para a eficiência do mesmo. Sendo assim, com o aumento na integração dos componentes, há um acréscimo também na produtividade, uma vez que todos os pesquisadores estão cientes do que está sendo realizado pela outra parte. Isto ocorre pelo fato de a documentação do trabalho ser criada, em grande parte, automaticamente. Ainda mais, os participantes não gastam muito tempo documentando seu trabalho, uma vez que a bibliografia já está de certa forma, encaminhada.

Tabela 3 - Sistemas

Classe	Sistema Escolhido
Controle de Versão	SubVersion
Gerenciamento de Requisitos	TRAC
Análise de Erros Runtime	Não decidido
Modelagem de Software	Não decidido

Conclusões

Portanto viu-se que se torna imprescindível o uso de ferramentas de controle durante a realização de projetos de engenharia. Constatou-se também que existem ferramentas open source poderosíssimas, com custo zero, que podem ampliar exponencialmente a velocidade na obtenção dos resultados. Sem contar que os resultados obtidos em projetos que fizeram o uso destas ferramentas serem mais confiáveis e facilmente interpretáveis, uma vez que a documentação destes projetos é mais detalhada.

Referências Bibliográficas

[1] RAFIK, Ouared; ALEC, Dorling; LOTHAR, Winzer; CASS, Carranza Juan Maria; Volcker. Spice for space trials, risk analysis, and process improvement. Software. Process Improvement and Practice., 9(1):13–21, 2004.

[2] COLLINGS-SUSSMAN, Ben e FITZPATRICK, Brian e PILATO, C. Michael. Version Control with Subversion [online] disponível na Internet via WWW. URL: http://svnbook.red-bean.com/en/1.4/svn-book.pdf. Acesso em 3 de Setembro de 2010

[3] CHACON, Scott. Git – Vast Version Control System. URL: http://git-scm.com/ Acesso em 10 de Setembro de 2010

[4] BERLIN, Daniel e ROONEY, Garret. Practical Subversion, Second Edition. Apress, 2006

[5] MURPHY, David. Managing Software Development with Trac and Subversion. Packt Publishing, 2007

[6] Trac-Hacks. URL: http://trac-hacks.org/ Acesso em 5 de Setembro de 2010

[7] LANG, Jean-Philipe URL: http://www.redmine.org/projects/redmine Acesso em 20 de Agosto de 2010