Ministério da Agricultura, Pecuária e Abastecimento

Comunicado Técnico 35

Dezembro, 2002
Campinas, SP

ISSN 1677-8464

Definição de um Processo de Software Leve

Marcos Lordello Chaim[1]

Como qualquer produto resultante de uma atividade de engenharia, é esperado que os produtos de software tenham características intrínsecas de qualidade perceptíveis pelo usuário. Segundo Rocha et al. (2001), a qualidade de software pode ser vista como um conjunto de características que devem ser alcançadas em um determinado grau para que o produto atenda às necessidades de seus usuários.

No entanto, observou-se que a qualidade do produto de software está ligada ao processo de geração do software. Em outras palavras, não é possível adicionar qualidade a um software depois de pronto. Assim, houve nos anos 90 uma grande preocupação quanto à qualidade do processo das organizações de desenvolvimento de software. Essa preocupação resultou em modelos para avaliar e melhorar os processos de software (e.g., *Capability Maturity Model* – CMM; *Software Process Improvement and Capability dEtermination* – SPICE), cujo objetivo é dar uma indicação da *maturidade* de um processo de software e definir ações para evoluí-lo.

Espera-se que durante os anos 2000 as organizações de desenvolvimento de software já tenham ajustado seus processos de software para a produção de produtos de qualidade dentro de prazos confiáveis (Rocha et al., 2001). Mais ainda, prevê-se que essas organizações serão pressionadas constantemente a otimizar os seus processos de desenvolvimento e manutenção, de forma a produzir produtos cada vez a custos menores e com qualidade crescente.

A principal atividade da Embrapa Informática Agropecuária é desenvolver pesquisa sobre o uso da tecnologia de informática no domínio agropecuário. Muitas vezes, como resultado dessa pesquisa, tem-se o desenvolvimento de produtos de software. Portanto, em larga medida, a Embrapa Informática Agropecuária é uma organização desenvolvedora de software e que também está sendo pressionada para adequar-se ao novo paradigma de eficiência e qualidade dos processos de software.

Assim, para preparar a Embrapa Informática Agropecuária para este desafio, está sendo realizado um esforço inicial para o estabelecimento de um processo de software cuja característica principal é ser repetível e gerar produtos com qualidade, custos e prazos de entrega previsíveis. A partir da observação das boas práticas correntes na organização e das experiências de sucesso no desenvolvimento de software (Cusumano & Selby, 1997; Beck, 2000; Raymond, 2001), optou-se pela implantação de um processo de software *leve* baseado nos seguintes princípios:

[1] *Engenheiro Elétrico, Mestre e Doutor em Engenharia Elétrica, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo – 13083-970 – Campinas, SP. (e-mail: chaim@cnptia.embrapa.br)*

Embrapa

- pouca burocracia e adaptação às características dos projetos;
- poucas diretrizes básicas – obrigatoriamente seguidas pelos projetos – que garantam o *gerenciamento dos projetos* e o *controle de configuração* das representações de software;
- utilização de uma linguagem comum para a definição das representações de software;
- uso de ferramentas de domínio público; e
- teste *freqüente* e *cedo* de todas as representações de software.

O objetivo desse documento é apresentar o processo de software leve da Embrapa Informática Agropecuária e as ações para implementá-lo. Na próxima seção, os princípios que norteiam o processo são detalhados. Em seguida, é apresentado o conjunto de ações visando a implantação do processo e os trabalhos relacionados. A última seção contém um resumo da proposta apresentada e os próximos passos.

Princípios do Processo de Software Leve

Pouca Burocracia e Adaptação às Características dos Projetos

A idéia subjacente a um processo de software leve é que ele deve impor o menor ônus burocrático possível aos desenvolvedores. O objetivo é deixar que eles façam aquilo que gostam e fazem melhor – desenvolver software. Todavia, pela própria característica do trabalho de desenvolvimento de software, a comunicação freqüente é necessária de forma que alguma burocracia envolvendo geração de documentos e realização de reuniões sempre será necessária.

No processo de software leve, caberá aos membros do projeto definir as representações de software (e.g., documentos, diagramas, código fonte, executável) que serão mantidas e controladas, bem como a forma de interação entre eles. A idéia é que haja muita comunicação informal (par a par) e pouca reunião.

Em resumo, pretende-se que o processo seja adaptável às características dos projetos. Uma vez definidos estes pontos – representações de software a serem utilizadas e forma de interação do projeto –, eles deverão ser seguidos e ser objeto de auditoria.

Diretrizes Básicas de Gerência de Projetos e de Configuração

Gerência de Projeto

O desenvolvimento de software é uma atividade de engenharia que tem como objetivo final a geração de um produto que possui as características esperadas pelo seu universo de usuários. Como características esperadas, pressupõe-se que o produto possua funções que atendam a uma demanda dos usuários de forma eficaz, eficiente e confiável. Além disso, é preciso que o produto seja obtido (com as características esperadas) dentro do orçamento e prazos definidos para seu desenvolvimento.

Para atingir esse objetivo, algumas tarefas essenciais são necessárias, a saber, estimar o custo de desenvolvimento do produto e controlar o andamento e execução do projeto de desenvolvimento. Estas atividades fazem parte da chamada *gerência de projeto*.

A rigor, estas atividades são necessárias para o desenvolvimento de qualquer produto de engenharia. Do ponto de vista de engenharia de software, a diferença é que os métodos de estimativa devem levar em consideração as características do produto de software (produto *lógico* no qual a qualidade é essencialmente introduzida durante o processo de desenvolvimento, visto que não há qualidade durante a fabricação) e as características do trabalho do desenvolvedor de software (trabalho basicamente intelectual que requer constante treinamento em novas tecnologias e ferramentas).

Gerência de Configuração

Como o produto de software é essencialmente obtido durante o processo de desenvolvimento, é fundamental que durante esse processo todas as representações do software (e.g., documentos, diagramas, código fonte, código executável, casos de teste) geradas sejam controladas.

Esse controle implica que as versões das representações do software são armazenadas e passíveis de serem recuperadas. Além disso, a gerência de configuração envolve o controle das configurações do software, isto é, o conjunto de versões das representações do software que define uma versão intermediária do produto.

Definição das Diretrizes Básicas e do Processo de Auditoria

De acordo com o princípio da pouca burocracia, o processo de software leve deve onerar o mínimo possível os desenvolvedores. No entanto, sem a definição de atividades de gerência de projeto e de configuração não se tem, de fato, um processo de desenvolvimento de software.

Assim, pretende-se definir um conjunto mínimo de diretrizes a serem seguidas pelos líderes de projeto de desenvolvimento de software da Embrapa Informática Agropecuária. O objetivo dessas diretrizes é garantir

que os projetos da unidade planejem atividades e escolham ferramentas para gerenciamento de projeto e de configuração. Além disso, para garantir que este conjunto mínimo de diretrizes para gerenciamento de projeto e configuração seja seguido, será definido um processo de auditoria dessas atividades nos projetos.

Por exemplo, com relação ao gerenciamento de projeto, as diretrizes deverão incluir a utilização de recursos clássicos de gerenciamento e acompanhamento de projetos como rastreamento de tarefas, gerenciamento de recursos, controle de custos e prazo etc., com a utilização de ferramentas automatizadas (e.g., MSProject (Microsoft Corporation, 2003), MrProject (Codefactory AB, 2003)).

É importante observar que esse princípio de pouca burocracia tem como base a cultura e o tipo de software desenvolvido na organização alvo. Isso porque o tipo de produto de software desenvolvido pela Embrapa Informática Agropecuária é de médio porte, em geral voltado para aplicações não-críticas, e desenvolvido por equipes perenes, não exigindo o controle burocrático requerido por produtos de grande porte direcionados para aplicações críticas.

Disseminação de uma Linguagem de Definição de Representações de Software

Software, à primeira vista, é um produto lógico que somente pode ser utilizado quando executado em um computador. Essencialmente, esse produto lógico, uma vez terminado o seu desenvolvimento, está pronto para ser liberado para o usuário.

No entanto, o programa em execução no hardware do usuário é apenas uma das representações do software. Outra representação é o código fonte que é compilado para gerar o programa executável. Antes do código fonte, podem existir outras representações não-executáveis do software: por exemplo, documentos de definição de requisitos, diagramas de análise, diagramas de arquitetura do software, diagramas de projeto, etc. Essas diferentes formas de representar um software definem uma linguagem para modelagem de sistemas de software.

Atualmente, a *Unified Modelling Language* (UML) (Booch et al.,1999) é o padrão *de facto* de linguagem de modelagem de sistemas de software baseada no paradigma de desenvolvimento orientado a objetos. Este padrão foi estabelecido pelo Object Management Group (2002) e é utilizado por diversas organizações desenvolvedoras de software.

Assim, pretende-se disseminar o padrão UML como a linguagem preferencial de desenvolvimento de representações de software da Embrapa Informática Agropecuária. A idéia não é tornar o uso dessa linguagem obrigatório, mas fornecer apoio a sua utilização, de forma que os desenvolvedores da Embrapa Informática Agropecuária passem a utilizar esse padrão para representar o software e também para comunicar entre si experiências, soluções, etc.

Uso de Ferramentas de Domínio Público

Paralelamente ao desenvolvimento da indústria de software, surgiu o movimento para desenvolvimento de produtos de software de domínio público. Este movimento tem como objetivo tornar disponível ferramentas de software sem custo. A idéia é fornecer alternativas viáveis aos produtos de software proprietários (Raymond, 2001).

Inicialmente, este movimento restringiu-se ao ambiente universitário com a produção de ferramentas para o desenvolvimento de programas (e.g., editores de texto, compiladores, etc.) para o sistema operacional Unix. Na década de 90, o movimento teve um impulso maior com o desenvolvimento de ferramentas de domínio público como o sistema operacional Linux, o navegador Netscape e o Servidor *Apache* para aplicações Web.

O movimento de software de domínio público também inovou na forma de desenvolver software; especialmente, a partir da popularização da Internet. Ao invés de contar com uma equipe de desenvolvimento de software dedicada ao desenvolvimento de um produto, os produtos de domínio público são desenvolvidos por grupos de voluntários que contribuem em tempo parcial. Normalmente, os voluntários desenvolvem novas funções e também testam as novas versões do software.

Para coordenar este esforço de vários desenvolvedores trabalhando de forma distribuída, foram desenvolvidas várias ferramentas de engenharia de software, também de domínio público. Por exemplo, a ferramenta para gerenciamento de projetos TUTOS (TUTOS..., 2002), o controlador de versões de arquivos CVS (Collabnet, 2002), o sistema para rastreamento e controle de defeitos Bugzilla (Mozilla Organization, 2002b), os ambientes integrados de desenvolvimento de programas como o NetBeans (NetBeans, 2002), etc. Algumas dessas ferramentas se mostraram tão eficientes que se tornam padrões da indústria como, por exemplo, as ferramentas CVS e Bugzilla.

Assim, o processo de software leve que se deseja implantar tem como objetivo aumentar o uso de ferramentas desse tipo na Embrapa Informática Agropecuária, visto que a Unidade já possui experiência nesse sentido, pois já utiliza várias delas, e também porque representam uma alternativa de baixo custo para a obtenção de apoio automatizado ao processo de software. A idéia é disseminar seu uso para apoiar as diretrizes identificadas no processo (gerenciamento de projetos e configuração) e a utilização do jargão UML.

No entanto, se não forem identificadas ferramentas de domínio público que realizem as funções desejadas com o desempenho desejado, ferramentas proprietárias poderão ser utilizadas.

Teste Freqüente e Cedo

Uma das atividades fundamentais de um processo de desenvolvimento de software é a garantia de qualidade. Esta atividade pode envolver revisões formais das representações do software, a verificação dessas representações utilizando métodos rigorosos e o teste propriamente dito do software.

Recentemente, tem-se observado um padrão repetido nas experiências bem sucedidas no desenvolvimento de software – a ocorrência freqüente e cedo da atividade de teste. Por exemplo, a MicroSoft (Cusumano & Selby, 1997) desenvolve seus produtos gerando versões diárias (chamados *builds*) e submetendo essas versões a um conjunto de casos de teste (chamado *smoke test*), de forma que, se algum teste falha, é realizada a correção imediata do defeito detectado. A vantagem dessa abordagem é que todo dia tem-se uma versão – que pode não ser ainda a que possui todas as funções desejadas – integrada e testada do software.

O paradigma de desenvolvimento de software de domínio público conhecido como *bazar* também possui um padrão de teste semelhante (Raymond, 2001). Nesse paradigma, versões (mesmo incompletas) do software são liberadas para uso; os usuários dessas versões, que em geral são também os próprios desenvolvedores voluntários, utilizam o software; se uma falha é observada, ela é imediatamente relatada e, muitas vezes, uma solução para o problema encontrado já é implementada pelo usuário/desenvolvedor. O coordenador do desenvolvimento do software de domínio público pode então ou incluir a solução proposta (se disponível) ou solicitar aos desenvolvedores voluntários a correção do defeito.

O paradigma de desenvolvimento *Extremme Programming* (XP) (Beck, 2000) utiliza as idéias acima no limite e no contexto de desenvolvimento de software proprietário. XP também preconiza a geração de versões intermediárias do software que vão incluindo incrementalmente funções definidas como prioritárias pelo futuro usuário do sistema. Para desenvolver estas versões intermediárias, são gerados testes unitários e teste de integração. Configurações integradas do software (*builds*) são geradas freqüentemente (em geral, mais de uma vez por dia) e todos os testes são executados. Se alguma falha é observada, ela é imediatamente corrigida, normalmente por um desenvolvedor diferente, para possibilitar que todos tenham conhecimento do código.

Note-se que nas experiências citadas o teste ocorre freqüentemente e cedo em versões intermediárias executáveis do software. Entretanto, não necessariamente toda representação de software é executável. Como então testar (cedo e freqüentemente) as representações de software não-executáveis? No contexto do processo de software leve, entende-se como o teste de representações não-executáveis a sua revisão formal.

Portanto, todo projeto deve definir como vai ser realizado o processo de teste freqüente e cedo. Esta definição deve prever como será conduzida a revisão das representações não-executáveis (e.g., documentos e diagramas gerados) e a revisão e teste das representações executáveis (e.g., código fonte, casos de teste). Tanto as revisões como os testes devem ocorrer freqüentemente, de forma que se tenha sempre uma versão intermediária do software revisada e testada.

Ações para Implantação do Processo de Software Leve

Para implantar o processo de software leve na Embrapa Informática Agropecuária, definiu-se um conjunto de ações com este objetivo:

Definição das Diretrizes Básicas e de Auditoria do Processo

Esta ação prevê a definição dos requisitos mínimos de gerenciamento de projeto e de configuração a serem auditados nos projetos. Além disso, é preciso definir um mecanismo de auditoria desses requisitos mínimos.

Identificação das Boas Práticas da Embrapa Informática Agropecuária

Uma das ações necessárias à implantação do processo de software leve é a identificação das boas práticas da Embrapa Informática Agropecuária. Por exemplo, a unidade possui uma experiência consolidada no controle de versões de arquivos utilizando a ferramenta CVS, no sistema para rastreamento e controle de defeitos Bugzilla, na utilização das ferramentas para geração automática de versões de software Tinderbox (Mozilla Organization, 2002c) e Bonsai (Mozilla Organization, 2002a), e na automatização da execução de casos de teste utilizando as ferramentas Junit (Object Mentor, 2002) e Dbunit (SourceForge, 2002), para teste unitário, e Vermont High Test (Vermont Creative Software, 2002), para o teste de integração.

No entanto, essas boas práticas não estão disponíveis para uso imediato dos novos projetos. Normalmente, para utilizá-las, é necessário um novo processo de aprendizado. Assim, nesta ação, pretende-se identificar as boas práticas já correntes na Embrapa Informática

Agropecuária e torná-las disponíveis para os novos projetos de uma forma simples e sistematizada.

Disseminação do Processo de Software Leve

A disseminação do processo de software leve dar-se-á de duas formas: utilizando um repositório de informações e por meio de cursos de um dia. A idéia é que haja um repositório de informações acessível via rede que contenha:

- as diretrizes básicas de gerenciamento de projeto e de configuração, bem como o mecanismo de auditoria;
- a descrição de técnicas de Engenharia de Software, em especial aquelas relacionadas com a realização das atividades de teste e revisão formal;
- a descrição das ferramentas de apoio ao processo de software leve, o treinamento básico para utilização e descrições de experiências de uso;
- e a descrição da linguagem UML, bem como de ferramentas de apoio a sua utilização e ao teste e revisão das representações geradas utilizando essa linguagem de modelagem.

A idéia é que o repositório tenha um formato fixo independentemente do conceito (ex.: diretriz, mecanismo, técnica, ferramenta, linguagem) que esteja sendo apresentado. O formato inicial proposto é descrito a seguir:

- O que é?
- Para que serve?
- Descrição:
- Exemplo de utilização:
- [Como derivar casos de teste:]
- Dicas ou *frequently asked questions*
- Links relacionados

Além disso, a disseminação do processo dar-se-á por meio de *cursos de um dia.* Nesses cursos, pretende-se apresentar a informação de um dado conceito contida no repositório em um único dia. Por exemplo, o curso de um dia do conceito de *Casos de Usos* da linguagem UML teria o seguinte formato:

Exemplo: Desenvolvimento de Casos de Uso

Manhã

O que é?

Para que serve?

Descrição: como desenvolver casos de uso;

Aplicação em um exemplo de utilização;

Tarde

Como derivar casos de teste a partir de casos de uso;

Descrevendo casos de uso com a ferramenta para modelagem UML;

Poseidon (Gentleware, 2002).

Trabalhos Relacionados

A definição e o futuro estabelecimento do processo de software leve visam aumentar o grau de maturidade da Embrapa Informática Agropecuária como organização desenvolvedora de software. O grau de maturidade da organização é estabelecido quando o seu processo de software é analisado utilizando modelos de avaliação e melhoria de processos tais como o *Capability Maturity Model* (CMM) e o *Software Process Improvement and Capability dEtermination* (SPICE). O modelo CMM, em especial, permite uma avaliação global do processo de software, enquanto o modelo SPICE avalia a maturidade de cada processo em particular do processo global de software.

O atual processo de software da Embrapa Informática Agropecuária pode ser avaliado, do ponto de vista do modelo CMM, como estando no nível *inicial*, isto é, a organização é capaz de produzir software de qualidade, mas o processo utilizado não é repetível e depende das habilidades dos membros da equipe de desenvolvimento.

No nível imediatamente superior do modelo CMM, o nível *repetível*, espera-se a existência de um sistema de gerenciamento que garanta o cumprimento de prazos, custos e níveis de qualidade. Nesse nível, o processo de software deve implementar as seguintes áreas-chaves: 1) gerenciamento de requisitos; 2) planejamento de projeto de software; 3) acompanhamento de projeto de software; 4) gerenciamento de subcontrato de software; 5) garantia de qualidade de software; e 6) gerenciamento de configuração.

Assim, observa-se que o esforço de estabelecimento do processo de software leve na Embrapa Informática Agropecuária visa implementar algumas das áreas-chaves citadas. O processo de software leve inova nos meios para implementação das áreas-chaves, pois baseia-se tanto em boas práticas relatadas na literatura (não relacionadas com melhoria de processos) como em boas práticas observadas no ambiente da própria organização. Com relação à automação das atividades das áreas-chaves, pretende-se utilizar ferramentas domínio público desde que seu desempenho seja adequado para a realização das atividades. Quando isso não for possível, poderão ser utilizadas ferramentas proprietárias.

Por exemplo, o estabelecimento de um padrão de linguagem de modelagem de sistemas (no caso a linguagem UML) tem como objetivo permitir o gerenciamento de requisitos. O planejamento e acompanhamento de projetos de software, por meio de um conjunto mínimo de diretrizes de gerenciamento de projetos, visa garantir a implementação dessas áreas básicas, porém, tendo como base a cultura e o tipo de software desenvolvido pela organização alvo. A idéia de utilizar-se ferramentas de domínio público para apoiar essas atividades é fornecer ferramentas a um custo baixo para apoiar o processo de software leve.

Finalmente, o conceito de teste *freqüente* e *cedo* foi estabelecido para garantir a qualidade dos produtos de software a partir da observação das boas experiências relatadas na literatura. O teste freqüente e cedo tem se mostrado uma técnica eficaz de desenvolvimento de produtos de software de qualidade. Isto tem ocorrido no desenvolvimento de produtos da MicroSoft, de domínio público e que utilizam o paradigma *Extreme Programming*.

No entanto, o conceito de teste é menos estrito do que aquele normalmente encontrado na literatura – em geral, associado à execução do programa por meio de casos de teste. A idéia aqui é estender esse conceito para toda a representação de software. Quando ela é executável (e.g., um programa), então desenvolve-se casos de teste que devem ser executados; porém, se a representação é não-executável, então a execução propriamente dita é substituída pela revisão da representação de software.

É importante observar que a implantação do processo de software leve irá contribuir para o aumento do nível de maturidade do processo de software da Embrapa Informática Agropecuária. Porém, o objetivo não é atingir um nível específico do modelo CMM, mas aumentar a maturidade em determinados processos, identificados como relevantes para a organização. De forma que, de acordo como o modelo SPICE, esses processos poderão possuir nível de maturidade maior que 1, mas não o processo de software como um todo.

Considerações Finais

Neste documento foi apresentada uma proposta de estabelecimento de um processo de software leve. A idéia do processo de software leve é impor o menor ônus burocrático para os desenvolvedores e ao mesmo tempo permitir à gerência controlar os projetos visando dar-lhes previsibilidade de custos e qualidade.

Este processo é baseado em cinco princípios:

- pouca burocracia e adaptação às características dos projetos;
- poucas diretrizes básicas – obrigatoriamente seguidas pelos projetos – que garantam o *gerenciamento dos projetos* e o *controle de configuração* das representações de software;
- utilização de uma linguagem comum para a definição das representações de software;
- uso de ferramentas de domínio público; e
- teste *freqüente* e *cedo* de todas as representações de software.

Ações para implantação desse processo foram definidas e, a partir delas, um cronograma de atividades visando a sua execução foi estabelecido.

Referências Bibliográficas

BECK, K. **Extreme programming explained**: embrace change. Boston: Addison Wesley, 2000. 224 p.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The unified modeling languague user guide**. Reading: Addison-Wesley, 1999. 482p. (Addison-Wesley Object Technology Series).

CODEFACTORY AB. **MrProject**. Disponível em: <http://mrproject.codefactory.se>. Acesso em: 16 jan. 2003.

COLLABNET. **Concurrent Versions System**. Disponível em: <http://www.cvshome.org>. Acesso em: 16 dez. 2002.

CUSUMANO, M. A.; SELBY, R. W. How Microsoft builds software. **Communications of the ACM**, v. 40, n. 6, p. 53-61, June, 1997.

GENTLEWARE AG. **Gentleware - tools for developers**. Disponível em: <http://www.gentleware.com/>. Acesso em: 16 dez. 2002.

MICROSOFT CORPORATION. **Microsoft Office - Microsoft Project**. Disponível em: <http://www.microsoft.com/office/project/default.asp>. Acesso em: 16 jan. 2003.

MOZILLA ORGANIZATION. **Bonsai**. Disponível em: <http://www.mozilla.org/bonsai.html>. Acesso em: 17 dez. 2002a.

MOZILLA ORGANIZATION. **Bugzilla - mozilla.org - the Mozilla bug database**. Disponível em: <http://bugzilla.mozilla.org>. Acesso em: 10 dez. 2002b.

MOZILLA ORGANIZATION. **Tinderbox**. Disponível em: <http://www.mozilla.org/tinderbox.html>. Acesso em: 17 dez. 2002c.

NETBEANS. **NetBeans [home page]**. Disponível em: <www.netbeans.org>. Acesso em: 16 dez. 2002.

OBJECT MANAGEMENT GROUP. **Object Management Group [home page]**. Disponível em: <http://www.omg.org>. Acesso em: 16 dez. 2002.

OBJECT MENTOR. **JUnit, testing resources for extreme programming**. Disponível em: <http://www.junit.org/index.html>. Acesso em: 16 dez. 2002.

RAYMOND, E. S. **The cathedral & the bazaar**: musings on Linux and open source by an accidental revolutionary. Beijing: O´Reilly & Associates, 2001. 241 p.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. (Ed.). **Qualidade de software**: teoria e prática. São Paulo: Prentice-Hall, 2001. 303 p.

SOURCEFORGE. **The Dbunit framework**: the Dbunit database testing framework. Disponível em: <http://dbunit.sourceforge.net>. Acesso em: 11 nov. 2002.

TUTOS – The ultimate team organization software. Disponível em: <http://www.tutos.org/>. Acesso em: 16 dez. 2002.

VERMONT CREATIVE SOFTWARE. **Vermont high test plus**: version 3.50 for Windows – the comprehensive testing tool that´s easy to use. Disponível em: <http://www.vtsoft.com>. Acesso em: 10 dez. 2002.

Comunicado Técnico, 35

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Av. André Tosello, 209
Cidade Universitária - "Zeferino Vaz"
Barão Geraldo - Caixa Postal 6041
13083-970 - Campinas, SP
Telefone (19) 3789-5743 - Fax (19) 3289-9594
e-mail: sac@cnptia.embrapa.br

1ª edição
2002 - on-line

Expediente

Supervisor editorial: *Ivanilde Dispato*
Normalização bibliográfica: *Marcia Izabel* Fugisawa Souza
Capa: *Intermídia Publicações Científicas*
Editoração Eletrônica: *Intermídia Publicações Científicas*