Variantes do Sistema

Håkan Dyrhage

Rational Software White Paper

TP 155

Rational®
the software development company

Índice Analítico

Introdução

Esse documento discute quais são as variantes de sistemas e como gerenciá-las. Você não precisa lê-lo para entender o Rational Unified Process,—pelo contrário, deve ser tratado como uma extensão para o Rational Unified Process. A última seção descreve de forma resumida como o Rational Unified Process seria afetado pela introdução de variantes e variabilidade.

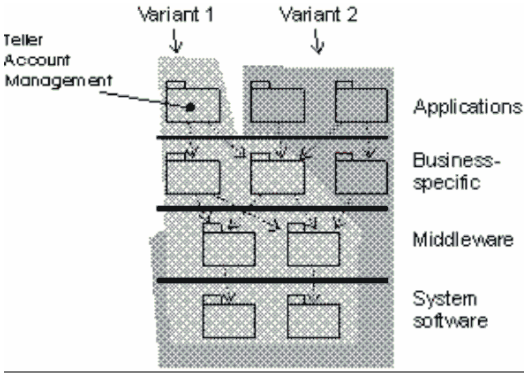
Essa é uma área em que o Rational Unified Process aprimorará e expandirá futuramente. Este documento oferece uma primeira amostra disso.

Variantes do Sistema

Muitos sistemas são fornecidos em mais de uma variante. Isso significa que o sistema é configurado, empacotado e instalado diretamente em diferentes (classes de) clientes. Às vezes, variantes diferentes são conseguidas simplesmente instalado-se e adaptando-se o sistema de maneira diferente. Outras vezes, a variabilidade é obtida distribuindo partes diferentes do sistema a clientes diferentes. As seções a seguir contêm alguns exemplos de variantes.

Partes Diferentes do Sistema

Partes diferentes do sistema completo são fornecidas a diferentes classes de clientes. Por exemplo, um sistema financeiro é fornecido como dois produtos diferentes. Vamos supor que a variante 1 do sistema contenha tudo sobre o setor financeiro de telefonia, ao passo que a variante 2 contém tudo sobre o gerenciamento de conta de caixa de banco. As executáveis são definidas nos subsistemas na camada de aplicativo. Isso significa que uma variante (variante 1 na figura a seguir) é criada, por exemplo, com o subsistema de Gerenciamento de Conta de caixa de banco, e todos os subsistemas ela precisa compilar e executar—ou seja, todos os subsistemas ela importa diretamente ou indiretamente.



Um sistema financeiro desenvolvido como duas variantes.

Idiomas Diferentes

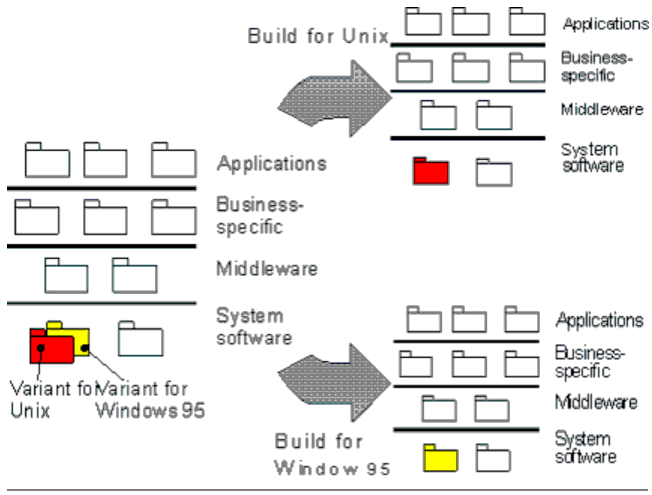
Se o sistema for produzido para idiomas diferentes, por exemplo, inglês, francês e japonês, será preciso fornecer uma variante do sistema para cada idioma. A diferença entre as variantes é que todo texto, como os menus e textos de ajuda, deve estar no idioma específico.

Uma maneira de lidar com idiomas diferentes é coletar todos os textos em um arquivo e ter um arquivo para cada idioma. Fornecer um sistema para um idioma específico significa que você fornece tudo, inclusive o arquivo que contém o texto para o idioma específico. Este arquivo é então lido pelo software no momento da inicialização e todas as variantes relevantes são inicializadas.

Plataformas Múltiplas

Se o sistema suportar várias plataformas incompatíveis, uma variante do sistema será necessária para cada plataforma. Por exemplo, se um sistema executar ambos, o Windows NT e o UNIX, duas variantes do sistema serão produzidas.

No exemplo a seguir, o código específico de plataforma está localizado em um subsistema. Neste caso, duas variantes do subsistema são desenvolvidas. Um arquivo de compilação—a "makefile"—especifica qual versão de cada código-fonte deve ser compilada junto. Você também pode dizer que um arquivo pronto especifica qual variante de cada subsistema deve ser parte do build. Portanto, você precisa de um arquivo pronto para cada plataforma.



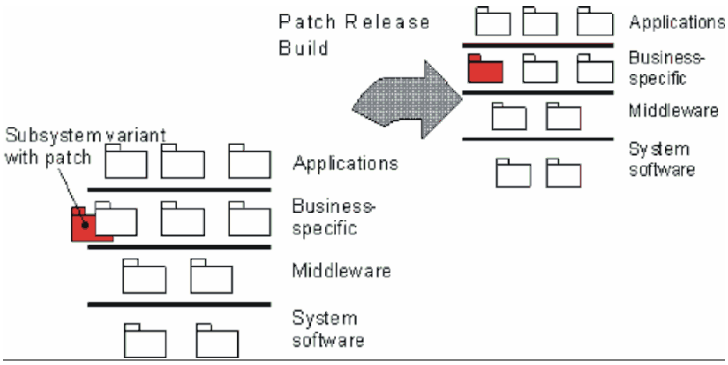
Um sistema é desenvolvido para executar em várias plataformas.

Liberação de Correção

Às vezes, é necessário desenvolver uma liberação de correção do sistema. Isso normalmente é feito em paralelo com um projeto de desenvolvimento, que significa que a liberação da correção é outra variante do sistema; ela existe em paralelo com a versão "main" do sistema.

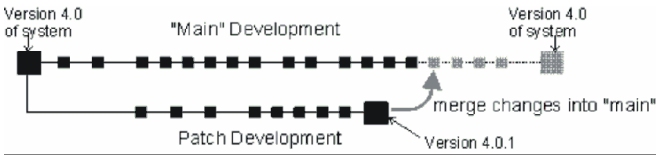
O desenvolvimento de uma liberação de correção é feito em paralelo com o projeto de desenvolvimento principal. Quadros cinzas indicam liberações futuras e linha de base.

As alterações necessárias estão localizadas em um ou em vários subsistemas de implementação. Como um projeto de desenvolvimento normal é feito em paralelo, você precisa desenvolver variantes desses subsistemas em paralelo com o desenvolvimento principal. Para criar um build, especifique em um arquivo pronto qual variante de cada subsistema deve ser parte do build.



Um build de liberação de correção é criado com uma variante do subsistema que contém a correção.

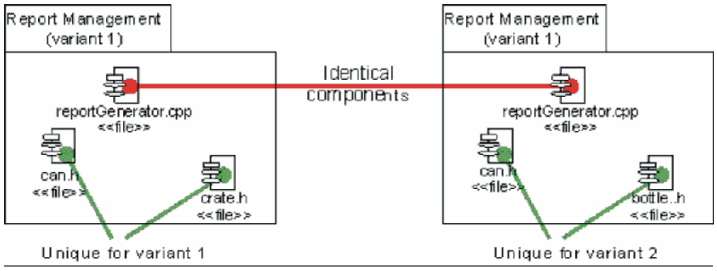
Normalmente, esse tipo de variante possui uma expectativa de vida breve. Após ser liberada a correção, essa variante não é desenvolvida mais e todas as alterações de código do valor são incorporadas na correção de desenvolvimento principal.



As alterações relevantes na liberação da correção são mescladas no caminho do desenvolvimento principal, quando conveniente.

Variantes dos Subsistemas

Os exemplos anteriores mostra que você, muitas vezes, precisa de duas ou mais variantes para um subsistema específico. Essas variantes terão alguns componentes em comum, ao passo que outros componentes são exclusivos para cada variante do subsistema.



Duas variantes de um subsistema de implementação chamado Gerenciamento de Relatório, onde foi decidido utilizar um componente como o mesmo em ambas as variantes. Os outros componentes são exclusivos para cada variante.

Freqüentemente, cada variante de um subsistema é desenvolvida por um implementador e as variantes do subsistema são desenvolvidas em paralelo. Se o componente for alterado por um implementador que desenvolveu uma variante, a alteração deverá ser propagada à outra variante. Para que isso seja possível, é preciso ter suporte à ferramenta de um CMVC (**Configuration Management and Version Control Tool**) para gerenciar os componentes que devem ser idênticos nas duas variantes.

Mecanismos para Variabilidade

Há vários mecanismos para criar variantes do sistema. Alguns deles foram mencionados nas seções anteriores. Cada mecanismo possui características diferentes. Normalmente, você utiliza uma combinação desses mecanismos para criar variabilidade em seu sistema.

- **Arquivos de compilação (e link)** . Especifique em um arquivo de compilação—um arquivo pronto em um ambiente Unix —, cuja variante de cada arquivo de código-fonte deve ser compilada e vinculada juntamente com os executáveis.
- **Componentes carregados dinamicamente**. Peças desenvolvidas do sistema como bibliotecas de link dinâmico, applets ou componentes do Active X que podem ser vinculados no programa em execução no tempo de execução. Esses componentes podem então ser gerenciados por uma ferramenta CMVC, que possibilita o fornecimento de subconjuntos deles a um cliente.
- **Arquivos de inicialização**. Utilizam arquivos que contenham informações que o software lê quando o sistema é iniciado para inicializar o sistema. Por exemplo, arquivos de recursos no Windows, arquivo de inicialização (startup ou initialization), que são lidos pelo software quando o sistema é iniciado, e definem diferentemente o sistema. Utilize isso, por exemplo, se o sistema for personalizado para idiomas diferentes, nos quais você mantém arquivos de texto que são lidos quando o sistema é iniciado.
- **Divida o sistema em vários executáveis**. Desenvolva o sistema como vários executáveis; por exemplo, arquivos .exe. As combinações deles podem ser fornecidas ao cliente. As diversas combinações de executáveis são gerenciadas por uma ferramenta CMVC.

Efeito no Rational Unified Process

Esta seção descreve de forma resumida, como o Rational Unified Process será afetado pela introdução de variáveis.

Efeito na Disciplina de Implementação.

A disciplina de implementação precisa ser ampliada nas seguintes áreas:

- Inclua as seguintes etapas na tarefa**Estruturar o Modelo de Implementação**:
  - Como definir quais variantes o sistema deve desenvolver
  - Como decidir quais subsistemas devem ter variantes
  - Como decidir qual mecanismo de variabilidade utilizar para obter variabilidade
- Na tarefa **Planeja a Integração do Sistema**, você precisa considerar as variantes e planejar como integrar diferentes variantes do sistema.
- Você precisa descrever mais detalhes sobre o desenvolvimento em paralelo "entre" variantes de um subsistema. Por exemplo, o que acontece se um componente "idêntico" precisar ser diferenciado ou como os componentes são mesclados? Isso também afeta várias das tarefas relacionadas com o implementador.

Outras tarefas de implementação e/ou atividades também podem ser afetadas.

Efeito em Outras Disciplinas

O desenvolvimento de variantes, ou famílias de sistemas, afeta todas as disciplinas. Nas seções anteriores, alguns efeitos na disciplina de implementação já foram discutidos. A seguir está uma breve lista de como as outras disciplinas serão afetadas:

- **Requisitos**—você deve descrever como identifica as variantes do sistema. Qual cada classe de cliente deseja.
- **Análise & Design**—deve descrever como você modela as variantes no modelo do design, como projeta as variantes dos subsistemas de design, e como define as variantes.
- **Teste**—deve descrever como testar uma família de sistemas (variantes). Teste cada variante do sistema como um sistema separado. Variantes também impactam no teste de integração.
- **Gerenciamento**—você precisa organizar o projeto, talvez em equipes separadas para cada variante do subsistema. Em um grande projeto, você pode até ter coordenadores de projeto separados para cada variante do sistema.
- **Ambiente**—você precisa de Ferramentas que ajudem a gerenciar as variantes dos sistema.
- **Implantação**—é muito mais complexa porque existem várias classes de clientes aos qual você oferecerá seu produto final.

Rational®
the software development company

Duas Sedes:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tel: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tel: (781) 676-2400

Sem custo: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com
Localização Internacional: www.rational.com/worldwide

Field Code Changed
Field Code Changed
Field Code Changed