



República Federativa do Brasil
Ministério da Economia
Instituto Nacional da Propriedade Industrial

(11) PI 0901035-1 B1

(22) Data do Depósito: 30/04/2009

(45) Data de Concessão: 31/03/2020



* B R P I 0 9 0 1 0 3 5 B 1 *

(54) Título: MÉTODO IMPLEMENTADO EM COMPUTADOR E SISTEMA BASEADO EM COMPUTADOR PARA A GERAÇÃO DE ARQUIVOS INTER-RELACIONADOS EXECUTÁVEIS EM COMPUTADOR

(51) Int.Cl.: G06F 8/34.

(52) CPC: G06F 8/34.

(30) Prioridade Unionista: 30/04/2008 EP 08 380135.7.

(73) Titular(es): ACCENTURE GLOBAL SERVICES LIMITED.

(72) Inventor(es): JORGE NICOLAU MOLINA; PEDRO OLMOS LÓPEZ; ALBERT JUANALS PASTOR; JOSE IGNACIO QUIROGA SERRET.

(57) Resumo: MÉTODO IMPLEMENTADO EM COMPUTADOR PARA A GERAÇÃO DE ARQUIVOS EXECUTÁVEIS EM COMPUTADOR INTER-RELACIONADOS, SISTEMA BASEADO EM COMPUTADOR E PRODUTO DE PROGRAMA DE COMPUTADOR. A presente invenção refere-se a um método implementado em computador para a geração de arquivos executáveis em computador interrelacionados em um lado de servidor, que compreende as etapas de: - provisão em um lado de cliente de uma interface gráfica de usuário associada a um código que pode ser lido em máquina em uma linguagem de máquina, em que a interface gráfica de usuário exibe dados substancialmente em texto claro; - seleção e associação de duas ou mais seções de código que pode ser lido em máquina em linguagem de máquina no referido lado de cliente usando-se a referida interface gráfica de usuário - verificação implementada em computador de uma coerência do referido código que pode ser lido em máquina selecionado no lado de cilindro antes, preferencialmente sem a compilação do referido código que pode ser lido em máquina selecionado no lado de cliente, e extração do resultado da referida verificação de coerência substancialmente em texto claro, em que: a verificação da coerência do referido código que pode ser lido em máquina selecionado compreende - a verificação implementada (...).

Relatório Descritivo da Patente de Invenção para **"MÉTODO IMPLEMENTADO EM COMPUTADOR E SISTEMA BASEADO EM COMPUTADOR PARA A GERAÇÃO DE ARQUIVOS INTER-RELACIONADOS EXECUTÁVEIS EM COMPUTADOR"**.

5 Descrição

A presente invenção refere-se a um método implementado em computador para a geração de arquivos executáveis em computador inter-relacionados, um sistema baseado em computador e um produto de programa de computador.

10 Convencionalmente, os programas de computador que podem ser aplicados em muitos campos são vendidos como produtos padronizados para um grande número de usuários. Esses programas de computador podem ser adaptados para rodarem em um sistema diretamente, isto é, sem a necessidade de uma rotina de instalação específica. Contudo, muito frequentemente, é necessário instalar um programa de computador em um sistema, antes de se rodar o programa de computador no sistema, em que, durante uma instalação, o programa de computador é adaptado ao ambiente de computação específico do respectivo sistema. Neste caso, tipicamente, o programa de computador é instalado por um usuário usando uma rotina de
15 instalação automática.

Adicionalmente a isso, dependendo das necessidades do usuário, os programas de computador podem ser tornados personalizados para um usuário específico. Convencionalmente, o usuário confia em um desenvolvedor ou uma equipe de desenvolvedores com o estabelecimento do código de computador personalizado, de acordo com as instruções do usuário. Contudo, de modo a se estabelecer esse código personalizado, um ou mais programadores altamente versados têm que ser empregados de modo a se criar o programa de computador. Este processo consome muito tempo é também é de custo muito intensivo.

30 É um objetivo da presente invenção prover arquivos executáveis em computador personalizados de uma maneira técnica simples. Este pro-

blema é resolvido pelos assuntos das reivindicações independentes. As modalidades preferidas são o assunto das reivindicações dependentes.

Método implementado em computador para a geração de arquivos executáveis em computador inter-relacionados de acordo com um aspecto

5 Um aspecto da presente invenção se refere a um método implementado em computador para a geração de arquivos executáveis em computador inter-relacionados em um lado de servidor, que compreende as etapas de:

10 - provisão em um lado de cliente de uma interface gráfica de usuário associada a um código que pode ser lido em máquina em uma linguagem de máquina, em que a interface gráfica de usuário exibe dados substancialmente em texto claro;

15 - seleção e associação de duas ou mais seções de código que pode ser lido em máquina em linguagem de máquina no referido lado de cliente usando-se a referida interface gráfica de usuário;

20 - verificação implementada em computador de uma coerência do referido código que pode ser lido em máquina selecionado no lado de cilindro antes, preferencialmente sem a compilação do referido código que pode ser lido em máquina selecionado no lado de cliente, e extração do resultado da referida verificação de coerência substancialmente em texto claro, em que:

a verificação da coerência do referido código que pode ser lido em máquina selecionado compreende

25 - a verificação implementada em computador da coerência das referidas seções selecionadas de código que pode ser lido em máquina e/ou
- a verificação implementada em computador da coerência das referidas associações das referidas seções selecionadas de código que pode ser lido em máquina; e

30 - mediante a aprovação da coerência, a compilação das duas ou mais seções selecionadas e associadas de código que pode ser lido em máquina em um lado de servidor para a criação de dois ou mais arquivos executáveis inter-relacionados, os quais são executáveis pelo servidor.

Vantajosamente, os arquivos executáveis em computador para um lado de servidor pode ser produzidos por uma pessoa, substancialmente sem quaisquer habilidades de programação em computador específicas. Ao invés disso, usando a interface gráfica de usuário (GUI) por meio de um texto claro, o usuário pode coletar duas ou mais seções de código que pode ser lido em máquina em linguagem de máquina sem ler esse código que pode ser lido em máquina ou a linguagem de máquina e, em particular, sem ter que conhecer um código que pode ser lido em máquina ou linguagem de máquina. O usuário pode escolher seções tais como "elementos", "validações", "textos/parágrafos"..., os quais são associados a seções específicas de código que pode ser lido em máquina em linguagem de máquina. Como um exemplo, uma validação pode ser associada a um código que pode ser lido em máquina compreendendo uma avaliação, quanto a se uma variável e/ou um parâmetro é maior do que uma outra variável e/ou parâmetro. As variáveis e/ou os parâmetros também podem ser especificados usando-se a GUI, em que o valor específico da variável e/ou do parâmetro é computado, quando o programa é executado.

Em outras palavras, o usuário pode criar substancialmente uma validação pela introdução de dois ou mais variáveis e/ou parâmetros e também ao introduzir o tipo de validação, por exemplo, uma operação tal como "maior do que", "menor do que", "igual a", etc. Mais ainda, o usuário pode introduzir a consequência da validação, isto é, o que será feito quando uma condição da validação for cumprida ou a referida condição não for cumprida. Como um exemplo, quando uma condição for cumprida, o método implementado em computador se moverá para uma próxima etapa.

Em outras palavras, usando a GUI, um código que pode ser lido em máquina complexo em linguagem de máquina pode ser montado, sem o usuário ter um conhecimento específico do código que pode ser lido em máquina ou de uma codificação em particular. Ao invés disso, o usuário pode ser uma pessoa tendo habilidades diferentes, tal como um engenheiro mecânico, um economista, etc., que precise usar os arquivos executáveis em computador e, portanto, deseje prover arquivos executáveis em computador

de acordo com suas necessidades. Consequentemente, há vantajosamente pouca perda de informação, uma vez que o usuário não tem que explicar para um desenvolvedor a necessidade dos arquivos executáveis.

Mais preferencialmente, a verificação implementada em computador de uma coerência do referido código que pode ser lido em máquina seletivo no lado de cliente antes da compilação do referido código que pode ser lido em máquina selecionado permite que o usuário de uma forma muito rápida e eficiente verifique a usabilidade dos arquivos executáveis produzidos em uma etapa posterior. Em particular, mesmo se não conectado a um lado de servidor e/ou se o lado de servidor não estiver disponível para o usuário por qualquer outra razão, o usuário poderá verificar já no lado de cliente, sem a produção dos executáveis, se os dois ou mais arquivos executáveis funcionarão apropriadamente no lado de servidor. Esta verificação pode ser feita, particularmente, embora o usuário não tenha habilidades de programação específicas, uma vez que a referida verificação da coerência é implementada em computador. Em outras palavras, no lado de cliente, o computador examina as seções selecionadas de código que pode ser lido em máquina. Desse modo, o computador pode examinar cada uma das seções separadamente. O computador também pode examinar as seções selecionadas em sua totalidade. Preferencialmente, o computador pode examinar qualquer inter-relação entre seções selecionadas de código. Como um exemplo, o computador pode examinar a saída de uma seção de código a qual seja introduzida em uma outra seção de código, como se o código fosse compilado.

Em outras palavras, o CPU pode examinar, sem compilar, se uma saída possível de uma primeira seção é adequada como entrada de uma outra seção. Portanto, também associações entre seções de código selecionado podem ser examinadas de uma forma computadorizada, pelo menos parcialmente automática. Desse modo, o computador pode aplicar regras específicas, tais como regras matemáticas e/ou uma lógica específica, tal como uma lógica matemática para o código que pode ser lido em máquina. Também, o computador pode usar regras e/ou uma lógica do código

que pode ser lido em máquina, tais como regras e/ou uma lógica de linguagem de codificação específica, de modo a examinar as seções selecionadas e/ou as associações entre as seções selecionadas do código.

Mais ainda, o computador no lado de cliente (isto é, um computador ou processador de cliente) pode examinar as seções selecionadas de código e/ou as associações quanto à inteireza. Como um exemplo, se uma seção estiver associada a uma outra seção, de modo que a saída de uma seção seja a entrada da outra seção, em que a outra seção precisa, por exemplo, de uma, duas ou mais variáveis de entrada, o computador poderá examinar qualquer incoerência, por exemplo, se a variável de saída de uma seção de código se adéqua às exigências da entrada da(s) outra(s) seção(s) de código. Mais ainda, o computador pode estimar o comportamento de tempo de rodada do código, mesmo sem uma compilação, por exemplo, pelo uso de lógica matemática. Como um exemplo, se o código compreender uma fração e o denominador da referida fração for uma variável a qual pode se tornar muito pequena durante um tempo de rodada, devido a razões matemáticas das faixas providas de variável/parâmetro, o computador no lado de cliente poderá descobrir que como resultado um estouro para cima de buffer ou similar ocorrerá no caso de o executável ser rodado no lado de servidor. Em particular, o CPU no lado de cliente pode reconhecer que o estouro de buffer para cima ocorrerá, embora sem a criação do executável (isto é, sem compilação) e mesmo sem a rodada do executável. Consequentemente, o computador pode prover uma respectiva saída de coerência.

A saída de coerência pode ser uma mensagem de erro ou uma mensagem de aprovação, particularmente em texto claro. Como um exemplo, os erros matemáticos ou similares, os quais podem ser providos quando da verificação da coerência do código selecionado e do composto em linguagem de máquina, tal como uma linguagem de programação, podem ser traduzidos em texto claro, em particular texto que pode ser lido por seres humanos, ainda mais particularmente de acordo com semântica legível por seres humanos e compreensível e regras de linguagem. Consequentemente, o usuário que não é um especialista em codificação, pode verificar se as se-

ções de código que pode ser lido em máquina, as quais foram selecionadas usando-se a GUI, podem ser rodadas de forma bem sucedida no lado de servidor, sem:

5 a) a necessidade de compilação das seções selecionadas do código no lado de servidor e/ou sem

b) a necessidade de o usuário ter habilidades de programação específicas, uma vez que a GUI provê uma saída legível e compreensível por um usuário não tendo habilidades de programação.

10 Como um exemplo, um tradutor pode ser provido conectado a e/ou usando uma ou mais tabelas compreendendo mensagens em lógica matemática e/ou lógica de linguagem que pode ser lida em máquina, etc., as quais são ligadas/associadas a um texto claro cobrindo o respectivo significado em linguagem legível por seres humanos/lógica.

15 O termo "texto claro" se refere a uma exibição de texto de acordo com semântica humana, gramática humana, linguagem humana, particularmente independente/diferente da linguagem que pode ser lida em máquina. Como um exemplo, um texto claro pode ser provido em Inglês, Espanhol, Alemão, etc. Um texto claro particularmente é um texto que pode ser lido e entendido por uma pessoa, sem ter habilidades específicas de programação,
20 como um exemplo sem ter habilidades específicas em qualquer linguagem de programação. Um texto claro também pode ser provido usando-se figuras, imagens, representações pictóricas, cores diferentes e/ou esquemas de cor, destaque de palavras, em particular uma exibição piscando de palavras, etc.

25 O termo "código que pode ser lido em máquina em linguagem de máquina" (contrário a texto claro) pode se relacionar a qualquer linguagem de programação convencional a qual pode ser, por exemplo, COBOL, FORTRAN, C, C++, PASCAL, JAVA, etc. Um código que pode ser lido em máquina em linguagem de máquina pode compreender elementos, os quais
30 também podem ser compreendidos por texto claro em linguagem que pode ser lida por seres humanos. Como um exemplo, um código que pode ser lido em máquina em linguagem de máquina pode compreender palavras de uma

linguagem humana, palavras em particular na língua inglesa, tais como "and", "if", "then", etc. Contudo, um código que pode ser lido em máquina em linguagem de máquina é baseado em lógica matemática, em particular como uma ferramenta matemática, se comparada com texto claro, que é baseada em gramática de uma linguagem humana como sua lógica intrínseca. Daí, o termo "linguagem de máquina" não está necessariamente limitado ao significado de uma linguagem de programação de nível baixo.

O termo "verificação implementada em computador de coerência" descreve o exame implementado em computador das seções de código selecionadas e/ou das associações entre as seções de código selecionadas, conforme descrito acima. Em particular, as seções de código selecionadas podem ser examinadas separadamente. Também é possível que conjuntos de seções possam ser examinados. Também é possível que a totalidade das seções selecionadas possa ser examinada. É possível que as associações entre duas ou mais seções de código sejam examinadas. As associações podem compreender, por exemplo, que a saída de uma seção seja a entrada de uma outra seção. As associações podem compreender, por exemplo, que duas ou mais seções compartilhem uma ou mais seções adicionais, variáveis/parâmetros, etc.

Contrário a "verificação implementada em computador de uma coerência", o termo "compilação" descreve a tradução de texto escrito em uma linguagem que pode ser lida em computador, a qual também pode ser referida com uma linguagem fonte, em uma outra linguagem que pode ser lida em computador a qual também pode ser referida como a linguagem-alvo. A sequência original usualmente é denominada o código fonte e a saída pode ser denominada um código de objeto. Comumente, a saída tem uma forma adequada para processamento por outros programas (por exemplo, por um ligante).

Pela compilação, um código fonte pode ser traduzido em um programa executável ou um arquivo executável. Em particular, um código fonte de uma linguagem de programação de nível alto pode ser traduzido em uma linguagem de nível mais baixo, por exemplo, tal como uma linguagem

assembly.

O termo "aprovação de coerência" descreve que uma verificação de coerência pode ser aprovada pelo computador cliente, em particular automaticamente, por exemplo, se um resultado específico for obtido pela verificação implementada em computador da coerência. É possível que, se um resultado específico não for obtido, uma mensagem de erro poderá ser extraída. Não obstante, mesmo no caso de uma mensagem de erro como essa, um usuário de forma alternativa ou adicional pode aprovar a coerência manualmente, por exemplo, se a saída da aprovação computadorizada prover um significado específico em texto claro, tal como, por exemplo, que certas variáveis não foram consideradas ou não foram consideradas apropriadamente pelos executáveis a serem gerados, etc.

O termo "arquivos executáveis inter-relacionados" descreve que dois ou mais arquivos executáveis, quando executados, funcionam em conjunto, por exemplo, compartilham recursos em comum, variáveis/parâmetros, etc. Também, o termo "arquivos executáveis inter-relacionados" pode descrever que um ou mais arquivos executáveis executam um ou mais outros arquivos executáveis. Também é possível que a saída de um ou mais arquivos executáveis seja a entrada de um ou mais arquivos executáveis. Desse modo, esses arquivos estão inter-relacionados.

O termo "interface gráfica de usuário está associada a um código que pode ser lido em máquina" descreve que a GUI tem uma relação com um código que pode ser lido em máquina, tal como acesso a um banco de dados, em que o código que pode ser lido em máquina é armazenado em tabelas, etc.

Seguindo-se a isso, de acordo com este aspecto, a geração de arquivos executáveis em computador inter-relacionados no lado de servidor pode ser facilmente realizada e/ou mantida. Em particular, vantajosamente, um código que pode ser lido em máquina em linguagem de máquina pode ser provido e composto de uma maneira muito fácil e suficiente por uma pessoa não sendo versada na técnica de processador. Mais vantajosamente, uma composição como essa de um código que pode ser lido em máquina

em linguagem de máquina pode ser obtida em um tempo muito curto, uma vez que, tipicamente, compilações muito longas não precisam necessariamente ser realizadas frequentemente. Ao invés disso, as seções selecionadas de código que pode ser lido em máquina precisam ser compiladas apenas uma vez, após uma aprovação bem sucedida da coerência. Nesse sentido, a verificação implementada em computador de uma coerência pode ser realizada de forma iterativa, particularmente, desde que até a coerência ser aprovada.

Em outras palavras, em um caso em que uma mensagem de erro é criada, usando-se a GUI, a seleção de duas ou mais seções de código que pode ser lido em máquina pode ser mudada e/ou as associações entre as duas ou mais seções de código que pode ser lido em máquina podem ser mudadas. A etapa de seleção e associação de duas ou mais seções de código que pode ser lido em máquina também pode incluir a regulação de variáveis/parâmetros usados pelas duas ou mais referidas seções de código que pode ser lido em máquina. Consequentemente, no caso de nenhuma aprovação da coerência já ser estabelecida, ao invés de se selecionarem e associarem seções adicionais de código que pode ser lido em máquina e/ou deixar de selecionar e/ou desassociar uma ou mais seções de código que pode ser lido em máquina, variáveis/parâmetros das duas ou mais seções selecionadas de código que pode ser lido em máquina podem ser mudados, apagados, regulados de novo, etc. É possível que a etapa de mudança, apagamento, reinicialização, etc. dos parâmetros/variáveis seja realizada iterativamente em combinação com a verificação implementada em computador de uma coerência, até uma aprovação bem sucedida da coerência.

A seleção e a associação de duas ou mais seções de código preferencialmente podem compreender um ou mais dos seguintes:

- a introdução de uma ou mais regras, tais como condições matemáticas, operações, etc., usando-se a GUI, em que uma ou mais referidas regras, tais como condições matemáticas, operações, etc., são introduzidas em texto claro;
- o armazenamento da entrada de uma ou mais tabela(s) de de-

finição específica(s);

- o armazenamento de uma ou mais regras, tais como condições matemáticas, operações, etc., em uma tabela de condições específica, preferencialmente diferente da(s) tabela(s) de definição;

5 - o armazenamento na(s) tabela(s) de definição e/ou na(s) tabela(s) de condição de um ou mais indicadores, tais como, por exemplo, por exemplo, ">", "<", "=>", "=<", "=", etc. indicando a natureza matemática de uma ou mais regras;

10 - a leitura, usando, por exemplo, um criador, a partir da(s) tabela(s) de definição e/ou da(s) tabela(s) de condição de uma ou mais regras e/ou de um ou mais indicadores;

- a provisão, por exemplo, usando-se o referido criador, em uma ou mais tabelas de código de um código em linguagem que pode ser lida em máquina;

15 - a geração, por exemplo, usando-se o referido criador, de um ou mais arquivos executáveis inter-relacionados a partir do referido código que pode ser lido em máquina. É possível que a partir de cada tabela de código um arquivo executável seja gerado. Também é possível que, a partir de uma tabela de código, dois ou mais arquivos executáveis sejam gerados.

20 Modalidades específicas do método implementado em computador

25 Como um exemplo, a seleção e a associação de duas ou mais seções de código que pode ser lido em máquina em linguagem de máquina compreendem a introdução de regras em texto claro usando-se a referida interface gráfica de usuário e provendo, por exemplo, usando-se o referido criador, um código que pode ser lido em máquina em linguagem de máquina equivalente ao referido texto claro, usando-se um gerador.

30 O termo "equivalente" no contexto do presente pedido compreende que, por exemplo, uma regra, condição matemática, etc., seja introduzida em texto claro (o qual pode ser facilmente entendido por um usuário que não seja versado em programação em uma linguagem de programação) e, então, seja transformada em linguagem que pode ser lida em máquina, a qual pode ser interpretada, por exemplo, por um computador.

Como um exemplo, a etapa de verificação implementada em computador da coerência compreende uma verificação implementada em computador da compatibilidade da primeira seção do código usado para a geração de um primeiro arquivo executável em uma segunda seção de código usada para a geração de um segundo arquivo executável.

Em outras palavras, vantajosamente, é facilmente possível examinar se dois arquivos executáveis funcionarão em conjunto apropriadamente, isto é, se a inter-relação dos dois referidos arquivos executáveis está correta, sem a necessidade de o usuário ter habilidades de programação específicas em um código que pode ser lido em máquina em linguagem de máquina.

Como um exemplo, a etapa de verificação implementada em computador da coerência compreende:

- com base no código que pode ser lido em máquina selecionado, a estimativa no lado de cliente de parâmetros de tempo de rodada, antes da compilação das seções selecionadas de código que pode ser lido em máquina, em que os parâmetros de tempo de rodada estão associados aos arquivos executáveis quando da execução dos arquivos executáveis no lado de servidor.

Em outras palavras, pela análise do código que pode ser lido em máquina selecionado, em particular pela análise da associação entre o código que pode ser lido em máquina selecionado, por exemplo, usando-se uma lógica de linguagem de programa e/ou uma lógica matemática, parâmetros de tempo de rodada podem ser estimados antes/sem uma compilação das seções selecionadas de código que pode ser lido em máquina. Os parâmetros de tempo de rodada podem ser variáveis/parâmetros usados por ou no(s) arquivo(s) executável(is) no lado de servidor. No caso em que os referidos parâmetros de tempo de rodada são incompatíveis com o sistema, em particular incompatíveis com as configurações de hardware no lado de servidor, uma respectiva mensagem pode ser gerada automaticamente e de forma computadorizada. Portanto, mesmo sem habilidades de programação específicas, o usuário pode verificar de uma maneira simples se o código

que pode ser lido em máquina selecionado funcionará apropriadamente no lado de servidor.

Como exemplos, a etapa de verificação implementada em computador da coerência pode compreender:

5 - Quando em uma lógica condicional (também referida como uma validação), um elemento definido como numérico é comparado com um elemento alfanumérico, uma mensagem de erro é provida, anunciando uma incompatibilidade.

10 - Quando um formato de um ou mais cálculos não está combinando/é compatível com o formato de seus operandos, uma mensagem de erro é provida. Como um exemplo, uma mensagem de erro pode ser provida quando a operação X é definida como Dec(6,2), isto é, uma variável tendo 6 entradas e 2 decimais e esta operação é o resultado da operação para somar variáveis Y e Z, em que Y é definido como Dec(15,2) e Z como
15 Dec(15,2). Em outras palavras, o resultado daquela soma "Y+Z" não se adaptaria ao elemento definido em X como "X = Y + Z".

Como um exemplo, uma ou mais das seções do código que pode ser lido em máquina são reusadas para a compilação de dois ou mais arquivos executáveis inter-relacionados.

20 Em outras palavras, seções similares ou idênticas de código que pode ser lido em máquina podem ser usadas para a compilação. A seleção de seções de código que pode ser lido em máquina pode compreender, portanto, a cópia das referidas seções. Contudo, embora seções similares ou idênticas de código que pode ser lido em máquina sejam usadas para a
25 compilação de arquivos executáveis diferentes, os referidos arquivos podem funcionar diferentemente, por exemplo, no caso em que as variáveis/os parâmetros são regulados diferentemente.

Em outras palavras, podem ser providos agrupamentos de código que pode ser lido em máquina, em que os referidos agrupamentos podem
30 ser unidos em conjunto pelo uso da GUI, de modo a se prover um conjunto de agrupamentos os quais podem ser compilados em um arquivo executável no lado de servidor. O referido conjunto de agrupamentos pode compreen-

der seções similares, particularmente idênticas de código que pode ser lido em máquina. Os agrupamentos também podem compreender conjuntos diferentes de código que pode ser lido em máquina. Em outras palavras, a GUI pode ser usada para a construção de um código que pode ser lido em máquina para compilação de dois ou mais arquivos executáveis, em que o código que pode ser lido em máquina para cada arquivo executável pode ser composto de forma similar a um kit de construção, em que o usuário usa o texto claro provido e introduz na GUI.

Como um exemplo, pelo menos um arquivo executável de primeiro nível é compilado, o qual está inter-relacionado a pelo menos um arquivo executável de nível i , em que pelo menos um arquivo executável de nível i está inter-relacionado com um arquivo executável de nível $(i+1)$, em que i é um número natural maior do que 1.

Como um exemplo, pelo menos um arquivo executável de nível i e pelo menos um arquivo executável de nível $(i+1)$ são compilados a partir de uma ou mais seções de um código que pode ser lido em máquina, em que uma ou mais seções de código que pode ser lido em máquina compreendem um código que pode ser lido em máquina similar, preferencialmente idêntico de forma substancial.

Em outras palavras, um conjunto hierárquico de arquivos executáveis pode ser criado, em que arquivos executáveis podem estar associados a arquivos executáveis, os quais, de novo, podem estar associados a arquivos executáveis adicionais. Conjuntos diferentes de arquivos executáveis podem ser criados, em que cada conjunto pode ter a referida estrutura hierárquica/de árvore. Cada nível pode ter um ou mais arquivos executáveis. Cada nível pode compreender arquivos executáveis, os quais podem ser similares, mesmo idênticos aos arquivos executáveis um ou mais níveis mais altos para cima na hierarquia ou um ou mais níveis mais baixos para baixo na hierarquia. Os arquivos executáveis em um nível podem ser inter-relacionados. Um ou mais dos arquivos executáveis em um nível pode ser inter-relacionados a um ou mais arquivos executáveis em um ou mais níveis hierárquicos mais altos e/ou um ou mais níveis hierárquicos mais baixos.

Diferentes conjuntos de arquivos executáveis podem ter uma estrutura hierárquica similar, mesmo idêntica. Diferentes conjuntos de arquivos executáveis podem ter estruturas hierárquicas diferentes. O valor de i pode ser 2, 3, 4, 5, 6, 7, 8, 9, 10, etc.

5 Como um exemplo, o método compreende a etapa adicional de verificação da integridade da execução de todos os arquivos executáveis pela execução de todos os arquivos executáveis no lado de servidor e pela comparação da performance no lado de servidor com uma especificação predeterminada no lado de cliente.

10 A performance no lado de servidor pode ser comparada de forma computadorizada com uma especificação predeterminada no lado de cliente. Como um exemplo, uma ou mais saídas dos arquivos executáveis no lado de servidor podem ser comparadas com um conjunto de variáveis/parâmetro providos no lado de cliente.

15 Como um exemplo, as etapas no lado de cliente são realizadas no computador de cliente remoto do servidor, e as etapas no lado de servidor são realizadas no computador de servidor remoto do cliente.

Em outras palavras, as etapas no lado de cliente são realizadas em um computador de cliente o qual pode ser um computador em uma rede.

20 O computador de cliente pode ser conectado a um computador de servidor através da rede. Após a realização das etapas no computador de cliente, em particular mediante uma aprovação da coerência, as duas ou mais seções selecionadas e associadas a um código que pode ser lido em máquina podem ser transferidas para o lado de servidor, em particular para o computador de servidor, por exemplo, através da referida rede. As duas ou mais seções selecionadas e associadas ao código que pode ser lido em máquina também podem ser transferidas para o servidor por qualquer outro meio, tal como pelo armazenamento das duas ou mais seções selecionadas e associadas de código que pode ser lido em máquina em um dispositivo de armazenamento, tal como um CD, uma unidade flash, uma unidade rígida, etc. e copiando-se as duas ou mais referidas seções selecionadas e associadas ao código que pode ser lido em máquina a partir do referido dispositivo de ar-

25

30

mazenamento para o servidor. No servidor, então, as duas ou mais seções selecionadas e associadas de código que pode ser lido em máquina então podem ser compiladas.

5 O dispositivo de armazenamento também pode fazer parte de um computador intermediário, o qual é conectado ao computador de cliente e ao computador de servidor. Em outras palavras, as duas ou mais seções selecionadas de código que pode ser lido em máquina podem ser transferidas a partir do computador de cliente para o computador de servidor por um outro computador ou um outro dispositivo computadorizado, tal como um
10 dispositivo móvel, para o servidor.

Alternativamente, as etapas no lado de cliente e as etapas no lado de servidor podem ser todas realizadas em um computador, o qual incorpora o servidor e o cliente.

Sistema computadorizado de acordo com um aspecto

15 Um aspecto se refere a um sistema computadorizado para a geração de arquivos executáveis em computador inter-relacionados em um lado de servidor, que compreende:

- uma interface gráfica de usuário em um lado de cliente, em que:
- 20 - a interface gráfica de usuário está associada a um código que pode ser lido em máquina,
- a interface gráfica de usuário exibe dados substancialmente em texto claro,
- a interface gráfica de usuário é adaptada para selecionar duas
25 ou mais seções de código que pode ser lido em máquina em um lado de cliente;
- um verificador de coerência o qual é adaptado para
- verificar uma coerência do referido código que pode ser lido em máquina selecionado no lado de cliente, e
- 30 - extrair o resultado da referida verificação de coerência substancialmente em texto claro; e
- um compilador para compilação das duas ou mais seções se-

leccionadas de código que pode ser lido em máquina em um lado de servidor para a criação de dois ou mais arquivos executáveis inter-relacionados os quais são executáveis pelo servidor.

Para a seleção de duas ou mais seções de código que pode ser lido em máquina em um lado de cliente, a GUI pode ser conectada a um seletor. O seletor pode compreender um ou mais links específicos, os quais apontam para seções de código em linguagem que pode ser lida em máquina. Em outras palavras, o seletor pode ser adaptado para selecionar uma ou mais seções de código, mediante uma entrada do usuário através da GUI, desse modo o usuário podendo se comunicar com a GUI em texto claro em linguagem que pode ser lida por seres humanos, e a GUI, através do seletor, pode selecionar duas ou mais seções de código representando o texto claro em código que pode ser lido em máquina. No contexto do presente pedido, um texto claro também pode compreender os campos de exibições de GUI, em particular botões e/ou outros tipos de menus, os quais são construídos, de modo que o usuário intuitivamente possa entender o significado dos referidos campos e/ou botões e/ou outro tipo de menus. Como um exemplo simples, de modo a se criar uma validação, o menu pode compreender botões tendo simbolizados neles operadores matemáticos, tais como ">", "<", "=", os quais usuário entenderá que representariam as operações em uma validação de "maior", "menor" e "igual a".

Tendo em vista o termo seções de código, os detalhes acima conforme realizados tendo em vista o método são igualmente aplicáveis.

Modalidades do sistema computadorizado

Como um exemplo, a interface gráfica de usuário compreende um gerador, o qual é adaptado para prover um código que pode ser lido em máquina em linguagem de máquina equivalente ao referido texto claro como uma entrada através da interface gráfica de usuário.

Como um exemplo, o sistema computadorizado pode compreender um estimador, o qual é adaptado para estimar no lado de cliente com base no código que pode ser lido em máquina selecionado os parâmetros de tempo de rodada, antes da compilação das seções selecionadas do código

que pode ser lido em máquina, em que os parâmetros de tempo de rodada são associados aos arquivos executáveis, quando da execução dos arquivos executáveis no lado de servidor.

Produto de programa de computador de acordo com um aspecto

5 Um aspecto do presente pedido se refere a um produto de programa de computador, armazenado em um dispositivo de armazenamento de dados ou provido como um sinal, o qual, quando carregado na memória de um computador e executado pelo computador obtém que o computador realize um método conforme descrito acima.

10 A descrição precedente dos aspectos não está limitada aos respectivos aspectos. Ao invés disso, a descrição dos respectivos aspectos pode ser aplicada aos aspectos remanescentes. Em particular, a descrição referente ao método pode ser aplicada para o sistema de computador e o produto de programa de computador e as respectivas modalidades específicas,
15 assim sendo, de modo que pela combinação de recursos individuais de aspectos específicos ou modalidades, novos aspectos e/ou modalidades possam ser criados. Mais ainda, os aspectos descritos acima e/ou modalidades não são restritivos a uma aplicação, tal como à geração de arquivos executáveis em computador inter-relacionados, os quais podem ser produtos de
20 seguro ou estar relacionados a produtos de seguro, e/ou relacionados a aplicação(ões) de gerenciamento de reivindicação de seguro, e/ou relacionados à geração de documentos de seguros, e/ou relacionados à geração de outros produtos, tais como produtos de imóveis, produtos de serviço financeiro e/ou documentos em um ou mais dos campos mencionados acima.

25 Descrição das Figuras

A seguir, exemplos específicos são descritos por meio de figuras de exemplo, em que recursos individuais das figuras podem ser combinados para outros exemplos, não explicitamente declarados na descrição das figuras. Com referência às figuras, é mostrado na:

30 Figura 1: uma visão geral esquemática de configurações de exemplo de seções de código que pode ser lido em máquina;

Figura 2: uma visão geral detalhada esquemática de um elemen-

to de acordo com a figura 1;

Figura 3: um instantâneo de uma GUI compreendendo um destaque;

Figura 4: um instantâneo compreendendo um destaque;

5 Figura 5a: um instantâneo compreendendo um destaque;

Figura 5b: um instantâneo compreendendo um destaque;

Figura 6: um instantâneo compreendendo um destaque;

Figura 7a: um instantâneo de uma GUI compreendendo um destaque;

10 Figura 7b: um instantâneo de uma GUI compreendendo um destaque;

Figura 8a: um instantâneo de uma GUI compreendendo um destaque;

15 Figura 8b: um instantâneo de uma GUI compreendendo um destaque;

Figura 9: um instantâneo compreendendo um destaque;

Figura 10a: um instantâneo compreendendo um destaque;

Figura 10b: um instantâneo compreendendo um destaque;

Figura 11a: um instantâneo compreendendo um destaque;

20 Figura 11b: um instantâneo compreendendo um destaque;

Figura 12a: um instantâneo compreendendo um destaque;

Figura 12b: um instantâneo compreendendo um destaque;

Figura 12c: um instantâneo compreendendo um destaque;

Figura 12d: um instantâneo compreendendo um destaque;

25 Figura 13: uma visão esquemática de um elemento de acordo com a figura 1;

Figura 14a: um instantâneo de uma GUI compreendendo um destaque;

30 Figura 14b: um instantâneo de uma GUI compreendendo um destaque;

Figura 14c: um instantâneo de uma GUI;

Figura 15: um instantâneo compreendendo um destaque;

- Figura 16a: um instantâneo compreendendo um destaque;
 Figura 16b: um instantâneo compreendendo um destaque;
 Figura 17a: um instantâneo compreendendo um destaque;
 Figura 17b: um instantâneo compreendendo um destaque;
 5 Figura 18a: um instantâneo compreendendo um destaque;
 Figura 18b: um instantâneo;
 Figura 18c: um instantâneo compreendendo um destaque;
 Figura 19: uma visão esquemática de um elemento de acordo
 com a figura 1;
- 10 Figura 20a: um instantâneo;
 Figura 20b: um instantâneo;
 Figura 20c: um instantâneo;
 Figura 21a: um instantâneo;
 Figura 21b: um instantâneo;
- 15 Figura 22: um instantâneo;
 Figura 23: uma visão geral esquemática que mostra a execução
 de dois ou mais arquivos executáveis inter-relacionados;
 Figura 24: um fluxograma;
 Figura 25: um instantâneo de uma GUI;
- 20 Figura 26: um instantâneo de uma GUI;
 Figura 27a: um instantâneo de uma GUI;
 Figura 27b: um instantâneo de uma GUI;
 Figura 27c: um instantâneo de uma GUI; e
 Figura 28: um sistema baseado em computador.
- 25 O método e o sistema descritos a seguir também podem ser re-
 feridos como "Info2000 Product Factory" ou "módulo de Info2000 Product
 Factory". Vantajosamente, o módulo de Info2000 Product Factory provê uma
 metodologia de desenvolvimento comum para todas as Linhas de Negócio,
 em que substancialmente todos os produtos, isto é, dois ou mais arquivos
 executáveis inter-relacionados são criados, suportados e manipulados da
 30 mesma forma, usando-se as mesmas telas e um dicionário de dados comum
 para armazenamento de dados. Mais vantajosamente, ele também permite

que um componente de produto extensivo reuso, tal como a reutilização de seções de código em linguagem que pode ser lida em máquina. No contexto deste pedido, o termo "reutilização" particularmente significa que uma seção de código específica pode ser copiada uma ou mais vezes para uso em uma

5 ou mais composições de código, as quais então são usadas para a criação dos dois ou mais arquivos executáveis inter-relacionados. Em outras palavras, o termo "seleção" no contexto deste pedido significa que a seção selecionada de código é copiada para a criação de dois ou mais arquivos executáveis inter-relacionados. Em particular, o termo "seleção e associação" no

10 contexto do presente pedido pode significar que o código é copiado em tabelas.

Uma vez que o Produto seja definido, isto é, uma vez que todas as seções de código que pode ser lido em máquina tenham sido selecionadas e associadas, possivelmente também todos os parâmetros/as variáveis

15 tenham sido regulados, o Product Factory gerará um programa executável compreendendo dois ou mais arquivos executáveis inter-relacionados, passíveis de serem chamados a partir de interfaces de Administração de Política.

1. Estrutura de Produto

20 A figura 1 provê um diagrama que mostra a estrutura básica de produto definida em Product Factory. Em outras palavras, a figura 1 mostra uma visão geral da funcionalidade dos dois ou mais arquivos executáveis inter-relacionados, uma vez que eles sejam executados no lado de servidor.

O desenvolvedor de produto definirá os diferentes componentes

25 que são necessários para a construção do Produto, tais como

- Objetos segurados (por exemplo, carro, casa)
- Coberturas (por exemplo, edificação)
- Códigos de Estimativa (para reivindicações; por exemplo, incêndio)

30 Estes componentes podem ser reusados dentre produtos. Daí, o desenvolvedor de produto pode selecionar os componentes usando uma GUI, onde na GUI os componentes estão disponíveis usando-se texto claro,

preferencialmente pelo fato de a GUI compreender respectivos botões, por exemplo, tal como um botão para "objetos segurados" (insured objects), os quais podem prover, por exemplo, uma lista suspensa de escolhas possíveis as quais então podem ser selecionadas, por exemplo, simplesmente por destaque, usando-se um dispositivo de apontar, como um mouse ou alguma coisa equivalente. Estes botões estão associados a um respectivo código que pode ser lido em máquina em linguagem de máquina.

Elementos também podem ser definidos e atribuídos no nível apropriado (por exemplo, ano de construção). Os elementos podem ser usados para a captura de dados ou para a regulação de regras. Os elementos podem ser simples campos de entrada ou estar relacionados a um conjunto de valores de referência, os quais podem ser mantidos ou armazenados em uma tabela de corporação.

Como um exemplo, na figura 1, o produto a ser criado se refere a um seguro de "propriedade de conforto de casa" ("House comfort property"). Em outras palavras, os arquivos executáveis inter-relacionados proverão, quando executados no lado de servidor, como resultado de documentos de seguro, tal como um contrato de um seguro de uma casa. O método de acordo com o presente pedido permite a geração computadorizada de dois ou mais arquivos executáveis inter-relacionados os quais, quando executados, criam ou ajudam quando da criação dos referidos documentos de seguros.

O método implementado em computador particularmente permite que uma pessoa não-versada na técnica de programação configure o código fonte de uma maneira simples e eficiente, de modo que dois ou mais referidos arquivos executáveis inter-relacionados possam ser criados no lado de servidor e possam ser executados no lado de servidor.

Em particular, os executáveis, quando executados, provêm um meio para a criação individual de documentos ao se permitir uma regulação através de uma GUI de todas as variáveis/os parâmetros, condições de limite, circunstâncias, elementos, etc.

De acordo com a figura 1, para o produto geral, bem como para

cada componente do produto, o desenvolvedor pode escolher um ou mais elementos, validações, operações e textos/parágrafos. Como um exemplo, o objeto do produto "House comfort property" pode ser uma "casa" (house). Adicionalmente, outros objetos poderiam ser introduzidos, tal como, por exemplo, "apartamento" (apartment).

Mais ainda, uma cobertura é selecionada, a qual pode ser "edificação" (building). Adicionalmente a isso, as coberturas "conteúdo" (contents), "roubo" (theft) e "ajudantes domésticos" (house aids) são selecionadas no exemplo de acordo com a figura 1. Para a cobertura "building", os códigos de estimativa "incêndio" (fire), "explosão" (explosion), "inundação" (flood), "raios" (lightning) e "terremoto" (earthquake) e o elemento "ano da construção" (building year) são selecionados.

2. Módulos de Product Factory

O Info2000 Product Factory compreende quatro módulos básicos, especificamente, definições de produto, validações, regras de classificação e texto/parágrafos.

- Definição de Produto

Este módulo permite que o desenvolvedor de produto configure a definição de produto, conforme descrito acima com respeito à figura 1. Toda a informação referente à definição de produto é armazenada no modelo de dados de Product Factory, em conjunto com a informação introduzida nos próximos módulos.

- Validações

Este módulo permite que o desenvolvedor de produto configure regras a serem executadas e validadas durante o ciclo de vida do produto em uma administração de política, isto é, quando da execução de dois ou mais arquivos executáveis inter-relacionados, de modo a se produzir, por exemplo, um documento de seguro. Uma validação possível pode ser definida, por exemplo, validar se um motorista é maior de 18 anos, quando executando os dois ou mais arquivos executáveis, em que uma variável é a idade de um motorista. O Product Factory permite definir variáveis/parâmetros, tal como a idade, quando da definição de uma validação, a qual será realizada

durante o tempo de rodada dos dois ou mais arquivos executáveis inter-relacionados (isto é, durante o tempo de rodada do produto). Estas validações como quaisquer outros componentes de produto são reusáveis dentre os produtos. Em outras palavras, a validação é para verificar se a idade de um motorista como uma variável de exemplo usada quando da definição, por exemplo, de um objeto do produto é maior do que um dado valor de limite. A referida validação também pode ser usada quando da definição de um objeto diferente do produto e/ou a cobertura e/ou o produto em si.

A figura 2 mostra uma visão geral de configurações possíveis de validação. Em particular, a figura 2 mostra a configuração possível de cada validação, eletrodo usando N condições lógicas, em que N pode ser um número natural, em particular 2, 3, 4, 5, 6, 8, 10, 15, 20, etc. Mais ainda, como um exemplo na figura 2 são mostrados os possíveis componentes de cada condição lógica. Cada uma das N condições lógicas pode compreender M operandos, em que M pode ser um número natural, em particular 2, 3, 4, 5, 6, 8, 10, 15, 20, etc. Mais ainda, para cada condição lógica, os M operandos lógicos podem ser conectados por um dos operadores lógicos. Consequentemente, são providos para cada condição lógica M-1 operadores lógicos. Os operandos lógicos podem compreender um ou mais dos seguintes: Elementos, Acesso, Operações, Funções pré-definidas, Tipo de objeto, Primeira classe, Constante numérica, Constante alfanumérica, Rotina, etc. O operador lógico pode compreender um ou mais dos seguintes: EQ = igual, NE = não-igual, GT = maior do que, GE = maior do que ou igual a, LT = menor do que, LE = menor do que ou igual a, TC = cobertura contratada, TNC = cobertura não contratada, TI = cobertura incompatível, TD = cobertura dependente de, etc.

As N condições lógicas podem ser conectadas por um operador booleano, tal como "AND", "OR", "(", ")", etc.

Também na figura 2 é mostrado um exemplo adicional de uma possível validação, especificamente, para se verificar se a idade de uma pessoal para quem os documentos de seguro são criados tem uma idade maior do que 100 anos. Se a referida validação descobrir que a idade, como

um parâmetro preferido, for maior do que 100 anos, uma ação específica será realizada, por exemplo, encargos mensais de 7,20 euros serão cobrados.

5 O módulo "Validações" pode ser acessado quando se usa o sistema de acordo com o presente pedido, seguindo-se o caminho *System>Prod.Conf.>Validations*. A partir daqui, novas validações podem ser adicionadas, modificadas ou apagadas.

A figura 3 mostra um instantâneo de uma interface gráfica de usuário (GUI) de exemplo do sistema baseado em computador de acordo com o presente pedido, quando da adição ou da modificação de uma validação, a qual foi criada para fins ilustrativos. Em outras palavras, o instantâneo de acordo com a figura 3 mostra uma validação a qual basicamente declara que a validação deve ocorrer se a soma dos conteúdos exceder ao número de 2.500.000. A parte de topo deste instantâneo pode conter o nome, a descrição, a data e os operadores. O retângulo destacado é em que a validação real é exibida. Este módulo será usado para validação dos dados de política como um gatilho de um cálculo dentro de uma operação ou como um gatilho de parágrafos. Se esta validação não ocorrer, então, ele abrirá uma mensagem de erro. Em outras palavras, a figura 3 mostra uma GUI 1 que exibe um texto claro, em particular tendo, por exemplo, os campos 2a, 2b, 4a, 4b, etc., para a introdução de texto, em que o usuário pode selecionar operadores específicos. Esta seleção através da GUI está associada a um código que pode ser lido em máquina em linguagem de máquina. Mais ainda, o usuário pode especificar variáveis e/ou parâmetros, os quais estão sujeitos, por exemplo, aos operadores. Através da GUI, as variáveis e/ou os parâmetros são regulados em linguagem que pode ser lida em máquina. Na figura 3, uma condição é exibida em texto claro. Em particular, os campos 2a, 2b são usados para a introdução de um período de tempo durante o qual a validação é aplicável. Como um exemplo, a data de começo introduzida através do campo 2a é escolhida na figura 3 como 1 de janeiro de 1900, e a data de fim introduzida através do campo 2b é 31 de dezembro de 9999. Durante este tempo, a validação será realizada, quando dois ou mais arquivos executá-

veis inter-relacionados forem executados.

O nome e uma descrição resumida da validação são introduzidos usando-se os campos 4a, 4b, em que os referidos arquivos são usados para se descrever brevemente a validação e, assim, permitir uma identificação simples da validação, por exemplo, na tabela de definição conforme mostrado na figura 4 (veja abaixo). Adicionalmente a isso, a GUI 1 de acordo com a figura 3 contém os campos 6a, 6b para a introdução dos operandos definindo a validação, bem como do operador no campo 8. A validação assim configurada e definida é mostrada na GUI 1, na janela 10, em que a linha que define a validação foi destacada por uma caixa 12 (não-mostrada quando da exibição da GUI) para uma visibilidade fácil. A GUI 1 compreende a definição de uma operação a qual é denominada OP_DEFAULT_8 e está usado a validação SUM CONTENTS 4 > LIM como um gatilho de cálculo 01 e 05, o que significa que o cálculo será realizado dependendo do resultado desta validação. Assim, dentro do cálculo, dependendo do resultado, C1 ou C2 poderia ser disparado. C3 sempre é disparado. Mais ainda, a descrição acima mantém, quando aplicável, uma similaridade com as GUIs, conforme mostrado nas outras figuras.

Como um exemplo, todas as validações são armazenadas na tabela KTPM34T. Em outras palavras, a tabela KTPM34T é uma tabela de definições de exemplo. A figura 4 mostra um instantâneo que compreende a definição da validação conforme selecionado com a GUI de acordo com a figura 3. São destacadas com uma caixa (não mostrada quando da exibição da GUI) as linhas contendo o código único de validação, neste exemplo "0000006619", datas válidas, neste exemplo, a partir de "19000101" a "99991231", as quais foram introduzidas através dos campos 12a, 12b da GUI 1 e da definição da validação. Também nas figuras a seguir, as caixas usadas para destaque normalmente não são exibidas pela GUI do sistema computadorizado.

A validação será executada apenas se a soma dos conteúdos for > 2.500.000. Estes valores sem os quais uma validação não será disparada são armazenados na tabela KTPA78T. Em outras palavras, a tabela

KTPA78T é uma tabela de condições de exemplo. As figuras 5a e 5b, as quais são dois instantâneos, destacam a condição definida com a GUI 1 de acordo com a figura 3 e também os referidos valores mencionados acima. Os instantâneos mostram duas linhas separadas definidas como R1 e R2, as
 5 quais não cabem em apenas um instantâneo, de modo que elas são exibidas nas figuras 5a e 5b.

Daí, os conteúdos são lidos a partir da tabela prévia tabela KTPA78T, um excerto da qual sendo mostrado nas figuras 5a e 5b, de modo que se uma soma for de além de 2.500.000, então, o programa seja execu-
 10 tado. Isto é o que o conteúdo da validação representa.

Em outras palavras, através da GUI 1 mostrada na figura 3, definições da validação são criadas em uma ou mais tabelas, no presente exemplo, entradas são criadas nas tabelas mostradas nas figuras 4, 5a e 5b, onde em uma tabela de definição (figura 4) a definição da validação em geral
 15 é provida e em uma outra tabela, isto é, na tabela de condição (figuras 5a e 5b), a condição em si e os valores das variáveis/dos parâmetros usados pela validação são definidos.

Um gerador tornado personalizado (fonte COBOL KTPLP007), de modo a se gerar a cópia em copybooks acessará aquela tabela KTPA78T, conforme mostrado nas figuras 5a, 5b, e executará as instruções a
 20 seguir representadas em pseudocódigo. A lógica a seguir em pseudocódigo representa a forma pela qual o programa opera e converte em fonte COBOL o que é configurado nas tabelas mencionadas:

Step 1: Read first record from KTPA78T (R1) and do

25 Step 2: Perform

Step 2.1: Write new line with the information of KTPA78T

Step 2.2: Read next record from KTPA78T

Until end of KTPA78T.

Step 3: Write new "End Lines".

30 O processo de acordo com as subetapas 2.1 e 2.2 é iterativo/repetido, isto é, o referido processo termina quando atinge o fim das linhas. No presente exemplo de acordo com as figuras 3, 4, 5a e 5b, duas

iterações serão realizadas.

Com referência à natureza matemática das regras inseridas em KTPA78T, na referida tabela os componentes são definidos dentro de validação(ões) e/ou operação(ões).

- 5 Como um exemplo, conforme mostrado nas figuras 5a e 5b, uma nova regra codificada por 000000619 é criada, a qual é uma CONDITIONAL LOGIC (LÓGICA CONDICIONAL) (daí, uma validação) pelo valor "CL" no campo TCTCP78T no qual o primeiro operando (campo CDORD78T com valor "01") e o operando é 0000006178 (campo CDCCH78T, o qual é uma
- 10 função preferida) deve ser maior do que (campo TCOBH78T com valor ">") do que o segundo operando (segunda linha na figura 5a) o que é numérico (campo TCTCH78T com valor N) e seu valor é 2500000.

- Assim, aqui a natureza matemática desta regra codificada é o campo TCOBH78T com o valor ">", onde o código que pode ser lido em máquina define que um valor derivado a partir da variável 0000006178 deve ser
- 15 comparado com um valor fixo, especificamente, 2.500.000.

Os indicadores possíveis podem ser ">", "<", "=>", "=<", "=", etc.

- O gerador tornado personalizado é projetado e adaptado para reconhecer condições específicas. Como um exemplo, a tabela KTPA78T
- 20 compreende no campo TCTCP78T o valor "CL", o que significa Lógica de Condição e é igual à operação "IF". Em outras palavras, como um exemplo, na(s) tabela(s) um ou mais parâmetros podem ser armazenados, os quais compreendem uma informação sobre condições lógicas. O gerador tornado personalizado é adaptado para ler o(s) parâmetro(s), identificar a lógica
- 25 do(s) parâmetro(s) e usar o(s) parâmetro(s) para criar um código fonte em cobol (ou um código fonte em qualquer outra linguagem que pode ser lida em máquina, tal como FORTRAN, C, C++, PASCAL, JAVA, etc.

- Como resultado, a validação é gerada em uma linguagem que pode ser lida em computador, por exemplo, em um código fonte em COBOL.
- 30 O instantâneo mostrado na figura 6 contém o código fonte do programa de resultado, o qual pode ser usado em uma etapa posterior quando da geração do(s) arquivo(s) executável (eis). Em outras palavras, a tabela mostrada

na figura 6 é uma tabela de código de exemplo, onde o código fonte é equivalente à condição, como uma entrada através da GUI mostrada na figura 3.

Portanto, usando o sistema baseado em computador de acordo com o presente pedido, o desenvolvedor de um produto, isto é, dois ou mais arquivos executáveis inter-relacionados, pode definir uma ou mais validações em texto claro, usando a GUI 1, conforme mostrado na figura 3. O conteúdo da GUI mostrado na figura 3 é transferido para uma tabela de definição, conforme mostrado na figura 4. Os valores numéricos e/ou as condições matemáticas da validação são armazenadas em uma tabela de condição, conforme mostrado nas figuras 5a e 5b. Usando as condições matemáticas e os valores numéricos, conforme armazenado na tabela de condição de acordo com as figuras 5a e 5b, um código fonte é automaticamente gerado. O código fonte é armazenado em uma tabela de código, conforme mostrado na figura 6. É possível que as condições geralmente sejam armazenadas em uma ou mais tabelas de condição e que valores numéricos, por exemplo, de variáveis/parâmetros usados nas condições armazenadas na(s) tabela(s) de condição sejam armazenados em uma ou mais tabelas de valor, e/ou uma ou mais tabelas de variável/parâmetro.

- Acessos

Este componente de Product Factory, isto é, o sistema/método baseado em computador do presente pedido para a criação de dois ou mais arquivos executáveis inter-relacionados, o qual é parte de um módulo, permite através de campos de chave uma busca de dados em uma tabela corporativa. Os acessos também são reusáveis dentre os produtos, isto é, diferentes métodos/sistemas baseados em computador compreendendo dois ou mais arquivos executáveis inter-relacionados.

É possível adicionar, modificar ou apagar um Acesso ao se seguir o caminho *System>Prod.Conf.>Accesses*. Os instantâneos conforme mostrado nas figuras 7a, 7b, 8a, 8b e 9 contêm o exemplo de um Acesso a uma tabela com todas as suas chaves e resultados econômicos. Cada Acesso tem um conjunto de chave(s) e resultado(s) associados. A seguir há um exemplo de um Acesso "ACC_SIC_2ND". As chaves são destacadas

nos dois instantâneos de acordo com as figuras 7a e 7b, como há muitas para caberem em um único instantâneo.

Os resultados econômicos mencionados anteriormente são destacados nos dois instantâneos de acordo com as figuras 8a e 8b.

5 Os acessos são armazenados na tabela KTPM38T. O instantâneo de acordo com a figura 9 contém a definição do exemplo acima em uma tabela de definição. São destacados os elementos relevantes, incluindo este código único "0000001280".

Os resultados econômicos deste Acesso, tais como dados econômicos relevantes os quais podem ser considerados quando da produção do código fonte e mais tarde nos arquivos executáveis, são armazenados na tabela KTPA30T. Estes são os mesmos que aparecem no exemplo do sistema/método implementado em computador, conforme mostrado pelas figuras. Eles são destacados nos instantâneos conforme mostrado nas figuras 10a e 10b, representando uma tabela de variável/parâmetro.

A tabela KTPA37T com as chaves para este Acesso em particular é exibida nos dois instantâneos de acordo com as figuras 11a e 11b, mostrando uma tabela de condição.

O código para o programa é exibido nos quatro instantâneos de acordo com as figuras 12a a 12d, que mostram uma tabela de código. Os resultados e as chaves são selecionados e, então, o Acesso é executado. O resultado é a geração de um Acesso.

O gerador (fonte de COBOL KTPLP007) de modo a gerar automaticamente o código em copybooks acessará ambas as tabelas (KTAP30T e KTPA37T) e executará as instruções a seguir representadas em pseudocódigo. As instruções a seguir em pseudocódigo representam como o programa opera e converte em uma fonte de COBOL o que é configurado nas tabelas mencionadas:

Step 1

30 Step 1.1 – Read first record from KTPA30T (R1) and do

Step 1.2 – Perform

Step 1.2.1 – Write new line with the information of

KTPA30T (L1)

Step 1.2.2 – Read next record from KTPA30T (R2)

Until end of KTPA30T.

Step 1.3 – Write new "Fixed Lines"

5 Step 2

Step 2.4 – Read first record from KTPA37T (R1) and do

Step 2.5 – Perform

Step 2.5.1 – Write new line with the information of

KTPA37T (L1)

10

Step 2.5.2 – Read next record from KTPA37T (R2)

Until end of KTPA37T.

Step 2.6 – Write new "Fixed Lines"

Step 3

Step 3.7 – Read first record from KTPA30T (R1) and do

15

Step 3.8 – Perform

Step 3.8.1 – Write new line with the information of

KTPA30T (L1)

Step 3.8.2 – Read next record from KTPA30T (R2)

Until end do KTPA30T

20

Step 3.8 – Write new "Fixed Lines" handling error, etc ...

No pseudocódigo acima, as "linhas fixas" (fixed lines) estão usando uma informação de código de tabela KTPM38T e código de acesso.

As subetapas 1.2.1 e 1.2.2 podem ser iterativas/repetidas. Também as subetapas 2.5.1 e 2.5.2 podem ser iterativas/repetidas.. Também as subetapas 3.8.1 e 3.8.2 podem ser iterativas/repetidas.

25

O pseudocódigo acima é refletido no código conforme mostrado nas figuras 12a a 12d, onde a figura 12a representa a etapa 1, a figura 12b representa a etapa 2 e as figuras 12c e 12d representam a etapa 3, a qual já começa na linha 061946 na figura 12b.

30

- Regras de Classificação

Através do módulo "regras de classificação", o desenvolvedor de produto define as operações (constituídas por um ou mais cálculos) usadas

para o cálculo da taxa de produto (os prêmios são calculados no nível de cobertura). Não apenas elementos podem ser usados em cálculos, mas valores (por exemplo, fatores) armazenados em tabelas corporativas podem ser usados através de "Acessos". Os programas de classificação externos
 5 podem ser "enganchados" nestes cálculos também. Uma visão geral esquemática é mostrada na figura 13, a qual é similar à figura 2. Em particular, a figura 13 mostra uma operação a qual pode compreender um grupo de cálculos, tais como n cálculos, em que n é um número natural. Em particular, n é igual a 1, 2, 3, 4, 5, 6, 8, 10, 15, 20, etc. Um cálculo pode ser uma vali-
 10 dação prévia compreendendo operandos e operadores, conforme descrito com referência à figura 3.

Os instantâneos de um exemplo gráfico são mostrados nas figuras 14a a 14c, os quais podem ser criados pelo exemplo do sistema/método implementado em computador, conforme mostrado pelas figuras, quando se
 15 segue o caminho *System>Prod.Conf.>Calculations*. A partir daqui, operações podem ser adicionadas, modificadas ou apagadas. Por exemplo, as figuras 14a a 14c representam uma composição de cálculos para uma operação denominada "OP_DEFAULT_8". Cada linha é destacada separadamente em uma respectiva figura. A primeira linha (na figura 14a) contém o
 20 cálculo "C1". A segunda linha (na figura 14b) contém o próximo cálculo "C2" e a última linha (na figura 14c) contém o último cálculo "C3".

No conteúdo acima dos cálculos, três níveis foram definidos:

- C1 (referido como a linha 01 na figura 14a) é um cálculo baseado em uma validação, a qual é rodada se a validação for FALSA (FALSE),
 25 isto é, o cálculo C1 é realizado se o resultado da validação "SUM CONTENTS 4 > LIM" for "falso". A validação "SUM CONTENTS 4 > LIM" foi definida como um exemplo com a figura 3 acima.

- C2 (referido como a linha 05 na figura 14b) é um cálculo baseado em uma validação "SUM CONTENTS 4 > LIM". O cálculo C2 é rodado
 30 se a referida validação for "verdadeira".

- C3 (referido como a linha 10 na figura 14c) é um cálculo que sempre é rodado.

A operação real é definida para o desenvolvedor de produto e armazenada na tabela KTPM56T, isto é, a tabela de definição. Esta tabela é onde todas as operações são armazenadas. A tabela KTPM56T é mostrada no instantâneo de acordo com a figura 15.

5 As definições deste cálculo estão contidas na tabela KTPA57T, como uma tabela de condição de exemplo, a qual é mostrada como um exemplo nos dois instantâneos de acordo com as figuras 16a e 16b. Para cada linha (R1, R2, R3), a qual é armazenada na tabela KTPA57T (figuras 16a e 16b), o formato do cálculo e quaisquer variáveis associadas são definidos.

10 O conteúdo do cálculo é armazenado na tabela KTPA78T, como uma tabela de variável/parâmetro de exemplo, a qual é mostrada como um exemplo nos dois instantâneos de acordo com as figuras 17a e 17b. Resumindo, a tabela KTPA78T mantém as relações matemáticas entre os operandos e os operadores, de modo que ao final a referida tabela armazene
15 como os elementos ou componentes são relacionados em uma regra matemática.

O código do programa é mostrado como um exemplo na tabela de acordo com os instantâneos nas figuras 18a a 18c, o que é uma tabela de código de exemplo. A partir das instruções a seguir em pseudocódigo pode
20 ser visto que o que acontece para cada linha. Como com validações e acessos, o gerador (fonte COBOL KTPLP007) gera um código automaticamente em copybooks que por sua vez acessarão as tabelas (KTPA57T e KTPA78T) e, então, informará estas instruções de pseudocódigo. Isto é como o programa torna o conteúdo de uma tabela em fonte cobol (em pseudocódigo):
25

STEP 1:

1.1-. Read first record from KTPA57T and write

1.2-. Perform

1.2.1- If there is a validation associated to the calculation:

30 1.2.1.1-. Write the PERFORM sentence to run the validation.

1.2.1.1-. Write the IF to evaluate the validation out-

come (True or False)

1.2.2.- Write the PERFORM to run the calculation. (C)

1.2.3.- If there is a validation associated to the calculation:

2.3.1.- Write End of lines and exit

5 1.2.2.- Read next record from KTPA57T

Until end of KTPA57T.

Como resultado do laço acima, o gerador gera o formato do cálculo com base no que é lido a partir de KTPA57T, e o código é gerado no copybook TP3CCYAU, conforme é mostrado no instantâneo de acordo com a figura 18a.

O cálculo de nível 01 (veja a definição dos níveis acima) é realizado, quando a validação for "falsa", então, o segundo nível 05 será realizado, quando a validação for "verdadeira", e o terceiro nível 10 será executado não sujeito a qualquer validação. Os níveis são indicados na figura 18a pelas respectivas caixas rotuladas "F" (cálculo de nível 01), "T" (cálculo de nível 05) e "Calculation" (Cálculo) (cálculo de nível 10).

Finalmente, também o conteúdo dos 3 níveis do cálculo (C1, C2 e C3) ou níveis 01, 05 e 10 é escrito no mesmo copybook TP3CCYAU. O pseudocódigo ou as instruções que o programa segue, de modo a se escrever o conteúdo, são conforme a seguir (pseudocódigo para acesso da tabela KTPA78T):

STEP 2:

2.1.- Read first record from KTPA78T and do

2.2.- Perform

25 2.2.1.- Write new line with the information of KTA78T

2.2.2.- Read next record from KTPA78T

Until end of KTPA78T.

Substeps 2.2.1 and 2.2.2 may be iterative/repeated.

Como resultado daquele laço (etapas 2.2.1 e 2.2.2), neste copybook mencionado o CONTENT destes cálculos (C1, C2 e C3 ou níveis 01, 05 e 10) também é escrito. No instantâneo de acordo com as figuras 18b e 18c, é mostrado como os cálculos são realizados em cada nível. Isto signifi-

ca que a PERFORM (EXECUTAR) (instrução em Cobol que executa o que está definido no interior) a qual é chamada na STEP 1 é definida aqui neste ponto na STEP 2, conforme mostrado nas figuras 18b e 18c.

- Textos/Parágrafos

5 O módulo "textos/parágrafos" permite a definição dos textos (modelos de documento) a serem gerados para cada produto, isto é, para cada conjunto de dois ou mais arquivos executáveis inter-relacionados, bem como as regras de disparo para os parágrafos (por exemplo, se uma cobertura A for selecionada, a oração XXX será impressa). Uma visão geral esquemática é mostrada na figura 19 (a qual é similar às figuras 2 e 13). Na
10 figura 19, a criação de acordo com o módulo de textos/parágrafos é mostrada. Como um exemplo, o "texto" pode compreender um número de parágrafos, tais como 1, 2, 3, 4, 5, 6, 10, 15, 20, etc. parágrafos, cada um dos quais podendo ser individualmente endereçado e individualmente associado, por
15 exemplo, a pelo menos um produto e/ou a pelo menos um objeto segurado e/ou pelo menos uma cobertura, etc. Na figura 19, o "Insured Object" está associado ao "Paragraph 1". Também é possível que 2 ou mais parágrafos estejam associados ao referido "Insured Object". Mais ainda, "Coverage 1" está associada a um parágrafo, especificamente "Paragraph 2". "Coverage
20 2" está associada a um parágrafo, especificamente "Paragraph 3". É possível que cada cobertura esteja associada a dois ou mais parágrafo e/ou que cada cobertura esteja associada a um parágrafo idêntico. Como um exemplo, "Coverage 1" pode ser associada a "Paragraph 1" e "Coverage 2" pode estar associada ao "Paragraph 2" e ao "Paragraph 3". Também é possível
25 que "Coverage 1" esteja associada ao "Paragraph 1" e ao "Paragraph 2".

 Como com as validações, os acessos e as operações, tabelas são usadas para a definição e o armazenamento de variáveis. A tabela KT-PA65T é a base para os documentos gerados para um produto.

30 Os parágrafos têm um código fixo, o que significa que naquela tabela há os parágrafos que estão associados a cada código de cobertura e também a cada produto, isto é, para cada conjunto de dois ou mais arquivos executáveis inter-relacionados.

Para o disparo de parágrafos, há também validações as quais estão associadas a eles, isto significando que um parágrafo de texto poderia ser disparado com base no fato que a cobertura a qual está associada é contratada e também se ela tem uma validação associada a ela.

5 Os instantâneos de acordo com as figuras 20a a 20c representam um exemplo desta definição de tabela KTPA65T.

Os instantâneos de acordo com as figuras 21a e 21b contêm a parte do programa em COBOL que acessa os campos de tabela KTPA65T e lê os valores para que se pretende que a documentação seja gerada. Como
10 resultado do acesso à tabela KTPA65T, o programa pode gerar a documentação requerida.

Aqui, o processo não é realmente o mesmo que nos outros níveis explicados, já que o executável que é chamado de modo a se dispararem os parágrafos (e, obviamente, documentos) lê os registradores associados a um parágrafo em particular e move toda a informação armazenada em
15 uma variável interna através de uma Instrução de Cobol denominada CURSOR, conforme mostrado nos instantâneos de acordo com as figuras 21a e 21b.

Finalmente, após a leitura da informação armazenada para cada
20 parágrafo, no mesmo copybook (executável) TPSTCDBD é checado quanto a se um certo parágrafo tem que ser disparado ou não, apesar do fato de ser contratado. Isto também é checado pelo código automatizado gerado com base nas validações associadas aos parágrafos os quais também foram transferidos (através de download) para aquele copybook, conforme mostrado no instantâneo de acordo com a figura 22.
25

3. Gerador de Produto

Após a configuração do produto usando-se os módulos prévios, isto é, após a composição de uma combinação de código que pode ser lido em máquina em linguagem de máquina pela seleção e associação de duas
30 ou mais seções de código usando-se a GUI, o Gerador de Product Factory transforma os referidos dados, os quais podem ser configurados a partir de tabelas, em arquivos executáveis em COBOL, que contém a base de um

produto, isto é, toda a informação de produto e as regras (exceto por aqueles valores mantidos em tabelas corporativas). No contexto do presente pedido, o termo "programa executável" pode ser sinônimo para o termo "arquivo executável" ou o termo "programa executável" pode compreender dois ou mais arquivos executáveis inter-relacionados. Através dos módulos de Administração de Política, políticas, validações de dados, etc. podem ser gerados para o produto.

Este Executável de Produto é chamado a partir de serviços de Administração de Política através de interfaces padronizadas, conforme mostrado como um exemplo na figura 23. De acordo com a figura 23, há um serviço de Administração de Política o qual é o KCILMZ06, o qual chama os executáveis quando o usuário ou quando a ação é VALIDATE (VALIDAR) ou RATE (CLASSIFICAR).

Ambas as ações podem ser disparadas nos três níveis diferentes:

- Política
- Objeto
- Cobertura

Agora, o serviço de Administração de Política sempre é o mesmo (KCILMZ06), mas os executáveis do Product Factory que realizam a ação de VALIDATING (VALIDAÇÃO) ou RATING (CLASSIFICAÇÃO) são dinâmicos. Isto significa que o serviço KCILMZ06 está acessando uma ou mais tabelas em que o nome do(s) executável(is) que precisa(m) ser usados em cada nível e em cada ação é armazenado. Esta tabela, por exemplo, é KTPM01T.

4. Sumário de Fluxos de Processo de Product Factory

A figura 24 mostra um fluxograma que resume os fluxos de processo principais descritos por todo o documento inteiro. A figura 24 descreve a funcionalidade de "Product Factory" especificamente para a geração em particular de programas de alta complexidade a partir de uma entrada inicial de dados.

As etapas S1 a S3 preferencialmente são realizadas no lado de

cliente. As etapas S4 a S6 são preferencialmente realizadas no lado de servidor.

Na etapa S1, os dados relevantes para a definição de dois ou mais arquivos executáveis inter-relacionados são introduzidos por um usuário do sistema/método baseado em computador de acordo com o presente pedido. O sistema/método baseado em computador também pode ser referido como um "Product Factory". Em particular, os dados são introduzidos usando-se uma GUI, conforme mostrado como exemplo nas figuras, tal como na figura 3.

Na etapa S2, os dados introduzidos são armazenados em uma ou mais tabelas, conforme também mostrado nas respectivas figuras, tal como nas figuras 4, 5a e 5b. As etapas 1 e 2 podem ser realizadas de uma maneira repetida, onde dados subsequentemente podem ser introduzidos (etapa 1) e os referidos dados podem ser armazenados em uma ou mais tabelas (etapa 2).

Na etapa S3, a verificação implementada em computador da coerência é realizada, onde uma ferramenta de coerência pode ser usada. A ferramenta de coerência pode compreender uma GUI para comunicação com o usuário do sistema. Em particular, a ferramenta de coerência pode ser adaptada para a extração de um resultado do teste de coerência em texto claro. Vantajosamente, a coerência do código em linguagem que pode ser lida em máquina pode ser verificada em sua totalidade no lado de cliente, particularmente sem uma compilação. Dois ou mais arquivos executáveis podem ser criados, então, no lado de servidor (veja as etapas S4 a S6) e rodados, com um risco minimizado de erros de tempo de rodada.

Na etapa S4, os processos compreendendo, por exemplo, validações, acessos, etc. podem ser gerados e, em particular, os dois ou mais arquivos executáveis inter-relacionados são compilados usando-se um compilador e, assim, na etapa S5 os dois ou mais arquivos executáveis inter-relacionados (referidos, por exemplo, como programas de resultado) são gerados.

Na etapa S6, dois ou mais arquivos executáveis inter-

relacionados são testados pela execução deles no lado de servidor.

A seguir, as vantagens do método/sistema implementado em computador de acordo com o presente pedido são descritas:

5.1 Evitar problemas de conexão

5 Esta vantagem será explicada por meio de um exemplo, onde uma operação 1 foi projetada conforme se segue:

Operação 1 = Elemento 1 + Elemento 2

Em um banco de dados específico, a Operação 1 pode ser conectada ao Technical Product (isto é, a cada conjunto de dois ou mais arquivos executáveis inter-relacionados) a que ela corresponder, de modo a se indicar para o sistema onde ele pode recuperar os valores para os componentes da operação (elemento 1 e elemento 2 neste exemplo). A conexão da operação significa que primeiramente os operandos devem ser conectados, nesse caso, o Elemento 1 e o Elemento 2, onde aqueles elementos devem ser atribuídos no nível desejado que poderia ser produto, objeto ou cobertura (os níveis são indicados na figura 1). Então, uma vez que os elementos sejam conectados, o sistema sabe onde os valores relacionados a eles são armazenados. O mesmo acontece com a operação. Como resultado desta operação, o fato de ser conectado significa que o sistema saberá onde os valores relacionados a ele estão armazenados.

Quando da geração do produto, isto é, dois ou mais arquivos executáveis inter-relacionados, o gerador verificará se todas as conexões foram eficientemente estabelecidas. Caso não, o processo falhará, e uma descrição de erro será escrita no registro de problemas.

25 De modo a se evitar uma falha, a operação pode ser apropriadamente conectada antes da geração. Mais ainda, de acordo com o presente pedido, a ferramenta de coerência é usada antes da geração, de modo a se descobrir se as conexões foram feitas eficientemente. Em um caso em que uma conexão incorreta foi feita, a ferramenta de coerência provê uma mensagem de erro, apontando a inconsistência e/ou uma conexão incorreta.

30 A ferramenta de coerência é a capacidade que permite checar (antes da geração) a coerência da definição dos componentes de Product Factory e suas

relações. Assim, esta ferramenta valida não apenas a definição, mas, também se as relações construídas também são consistentes. Em outras palavras, a ferramenta de coerência pode verificar se validações, elementos e operações estão corretamente correlacionados e oferecem integridade. A

5 ferramenta de coerência é adaptada em particular para ser executada usando-se a GUI. Como um exemplo, um usuário pode definir e criar um ou mais conjuntos de código fonte, usando a GUI, conforme mostrado na figura 25. Pela seleção do botão 16, uma etapa de projeto ou de associação de seleção e associação de duas ou mais seções de código que pode ser lido em

10 máquina inter-relacionado é realizada. Vantajosamente, um ou mais projetos/produtos podem ser escolhidos a partir da lista de produtos pré-definidos disponíveis. De acordo com o exemplo mostrado na figura 25, o modelo "Core_Motor" é selecionado. O modelo "Core_Motor" pode representar uma definição de duas ou mais seções selecionadas de código que pode ser lido em

15 em máquina. Pela seleção do botão 16, o código fonte do modelo "Core_Motor" é criado, em particular, as associações das duas ou mais seções de código que pode ser lido em máquina são estabelecidas.

Em uma próxima etapa, usando-se a referida GUI, a qual agora provê um botão 18, conforme mostrado na figura 26, a coerência é ativada, a

20 qual agora checa as duas ou mais seções selecionadas e associadas de código que pode ser lido em máquina, isto é, a verificação implementada em computador das associações de dois ou mais códigos que podem ser lidos em máquina selecionados e associados é realizada. Em outras palavras, a coerência das duas ou mais seções selecionadas e associadas de código

25 que pode ser lido em máquina é verificada usando-se a ferramenta de coerência pela seleção do botão 18. A ferramenta de coerência pode ser executada no fundo e prover uma mensagem, por exemplo, quando a verificação de coerência é negativa, particularmente provendo detalhes da natureza da coerência negativa. A ferramenta de coerência também pode prover uma

30 mensagem positiva, no caso de uma verificação positiva da coerência.

Também é possível que a ferramenta de coerência proveja uma janela individual na GUI ou seja exibida como uma janela na GUI, conforme

mostrado nas figuras 27a, 27b e 27c. Conforme mostrado nestas figuras, parâmetros específicos podem ser selecionados, tais como "Coerência de Elementos (Elements coherency)", "Coerência de validações" (Validations coherency), e/ou "coerência de classificações" (ratings coherency). Também, a GUI/ferramenta de coerência pode ser adaptada para prover detalhes da validação de coerência, conforme também mostrado nas figuras 27a, 27b e 27c.

5.2 Evitar problemas de formato

Dois exemplos são dados de modo a se descrever esta vantagem:

a) Validações

É possível ter a validação a seguir definida:

'If E1 > Op1 then'

em que:

E1 = valor de elemento

Op1 = resultado de um cálculo

Se o elemento E1 for definido com 'formato alfanumérico' (alphanumeric format) e o resultado da Operação 1 é definido como 'formato numérico (numeric format)', o gerador não será capaz de comparar seus valores e, conseqüentemente, nenhuma mensagem de erro será criada pelo gerador gerando os arquivos executáveis inter-relacionados. No tempo de rodada, o processo falhará e uma descrição de erro será escrita no registro de problemas.

Vantajosamente, a ferramenta de coerência pode verificar as inconsistências e extrair uma respectiva mensagem, mediante o que a inconsistência pode ser corrigida. No exemplo, E1 será definido como 'Numérico' (Numeric). Em outras palavras, mesmo se o gerador não prover qualquer aviso e/ou mensagem de erro, a ferramenta de coerência será capaz de determinar a inconsistência e extrair em texto claro um aviso e/ou uma mensagem de erro, mediante o que a validação pode ser corrigida.

Em outras palavras, a oração se compreendendo duas variáveis é uma seção de código. Pela verificação quanto a se as variáveis "E1" e

"Op1" podem ser comparadas usando-se o operador matemático ">", a coerência da seção de código é verificada, em particular se a referida seção de código é capaz de produzir uma saída consistente. Esta coerência pode ser particularmente verificada usando-se a ferramenta de coerência.

5 b) Operações

É possível ter a operação a seguir definida:

Operação 1 = Cálculo 1 + Cálculo 2

Cálculo 1 = Elemento 1 + Elemento 2

Cálculo 2 = Operação 2

10 em que :

Operação 1 é definida como: Dec (15, 3)

Cálculo 1 é definido como: Dec (15, 3)

Cálculo 2 é definido como: Dec (11, 6)

Operação 2 é definida como: Dec (15, 3)

15 Normalmente, esses resultados de operações que são importações econômicas, são definidos como Dec (15, 3), enquanto as percentagens são definidas como Dec (11, 6).

Neste exemplo, a Operação 2 é definida como Dec (15, 3), mas o cálculo 2 é definido como Dec (11, 6). A operação 2 é definida com um
20 formato numérico mais alto do que Cálculo 2. Neste caso, o gerador geraria um 'erro de aviso'. A geração e a compilação terminarão, mas o usuário será avisado quanto a este fato.

Durante o tempo de execução (também referido como tempo de rodada), se todos os resultados da operação 2 se adaptarem no formato definido pelo Cálculo 2, nada acontecerá. Mas há um risco no caso de resultados da operação 2 não se adaptar no formato de Cálculo 2 (por exemplo, 123456789, 122). Neste caso, durante um tempo de execução, o sistema
25 gerará um erro.

Vantajosamente, de modo a se evitar um erro possível durante
30 um tempo de rodada, o formato do Cálculo 2 poderá ser mudado para Dec (15, 3), onde o formato (15, 3) significa que o valor pode ter um máximo de 3 decimais e 12 inteiros; o formato (11, 6) significa que o valor pode ter como

máximo 6 decimais e 5 inteiros. Particularmente usando-se a ferramenta de coerência, o aviso criado pelo criador/compilador será traduzido e extraído em texto claro para o desenvolvedor, que não precisa conhecer a linguagem que pode ser lida em máquina, em que o gerador também extrai o aviso.

- 5 Consequentemente, o usuário também entende o aviso e pode adaptar o sistema pela correção da inconsistência. Em outras palavras, de acordo com o presente pedido, a coerência das associações das seções de código vantajosamente é verificada, onde a "Operação 1" representa uma primeira seção de código e a "Operação 2" representa uma segunda seção de código e
- 10 a associação entre as duas seções de código é realizada pela definição de:

$$\text{Operação 1} = \text{Cálculo 1} + \text{Cálculo 2}$$

em que "Cálculo 2 = Operação 2".

- Também é possível que "Operação 1" seja usada para a geração de um arquivo executável. "Operação 2" pode ser usada para a geração
- 15 de um outro arquivo executável. Consequentemente, ambos os arquivos executáveis são inter-relacionados, uma vez que a saída de um arquivo executável (por exemplo, a saída de "Operação 2" é usada como uma entrada para o outro arquivo executável (por exemplo, de "Operação 1").

5.3 Evitar problemas de sistema

- 20 Vantajosamente, os parâmetros de compilador são adaptados e otimizados, e o programa de classificação pode ser dividido, de modo que o processador, mesmo quando tiver uma capacidade limitada (outros processos possíveis podem estar rodando ao mesmo tempo), os arquivos executáveis inter-relacionados podem ser gerados sem falha da geração. Em parti-
- 25 cular, o gerador pode ser dividido em programas diferentes atendendo às funcionalidades (Operações, Validações, textos e documentação), desse modo se provendo uma geração confiável de arquivos executáveis inter-relacionados.

- Com referência à figura 28, um sistema de exemplo para a im-
- 30 plementação da invenção inclui um dispositivo de computação de finalidade geral na forma de um ambiente de computação convencional 20 (por exemplo, um computador pessoal) incluindo uma unidade de processamento 22,

uma memória de sistema 24 e um barramento de sistema 26, que acopla os vários componentes de sistema incluindo a memória de sistema 24 à unidade de processamento 22. A unidade de processamento 22 pode realizar operações aritméticas, lógicas e/ou de controle pelo acesso à memória de sistema 24. A memória de sistema 24 pode armazenar informação e instruções para uso em combinação com a unidade de processamento 22. A memória de sistema 24 pode incluir uma memória volátil e não volátil, tal como uma memória de acesso randômico (RAM) 28 e uma memória apenas de leitura (ROM) 30. Um sistema de entrada/saída básico (BIOS) contendo as rotinas básicas que ajudam na transferência de uma informação entre elementos no computador pessoal 20, tal como durante a partida, pode ser armazenado na ROM 30. O barramento de sistema 26 pode ser qualquer um de vários tipos de estruturas de barramento, incluindo um barramento de memória ou um controlador de memória, um barramento periférico e um barramento local usando qualquer um de uma variedade de arquiteturas de barramento.

O computador pessoal 20 ainda pode incluir um drive de disco rígido 32 para leitura a partir de e escrita em um disco rígido (não mostrado), e um drive de disco externo 34 para a leitura a partir de ou escrita em um disco removível 36. O disco removível pode ser um disco magnético para um driver de disco magnético ou um disco ótico tal como um CD ROM para um drive de disco ótico. O drive de disco rígido 34 e o drive de disco externo 34 são conectados ao barramento de sistema 26 por uma interface de drive de disco rígido 38 e por uma interface de drive de disco externo 40, respectivamente. Os drives e seus meios que podem ser lidos em computador associados provêem um armazenamento não volátil de instruções que podem ser lidas em computador, estruturas de dados, módulos de programa e outros dados para o computador pessoal 20. As estruturas de dados podem incluir dados relevantes da implementação do método, conforme descrito em maiores detalhes acima. Os dados relevantes podem ser organizados em um banco de dados, por exemplo, um banco de dados relacional ou de objeto.

Embora o ambiente de exemplo descrito aqui empregue um dis-

co rígido (não mostrado) e um disco externo, deve ser apreciado por aqueles versados na técnica que outros tipos de meios que podem ser lidos em computador os quais podem armazenar dados que são acessíveis por um computador, tais como cassetes magnéticos, cartões de memória flash, discos de vídeo digital, memórias de acesso randômico, memórias apenas de leitura, e similares, também podem ser usados no ambiente operacional de exemplo.

Vários módulos de programa podem ser armazenados no disco rígido, no disco externo, na ROM 30 ou na RAM 38, incluindo um sistema operacional (não mostrado), um ou mais programas aplicativos 44, outros módulos de programa (não mostrados) e dados de programa 46. Os programas aplicativos podem incluir pelo menos uma parte da funcionalidade conforme detalhada nas Figura 1 a 24.

Um usuário pode introduzir comandos e uma informação, conforme discutido abaixo, no computador pessoal 20 através de dispositivos de entrada tais como o teclado 48 e o mouse 50. Outros dispositivos de entrada (não mostrados) podem incluir um microfone (ou outros sensores), joystick, gamepad, scanner ou similares. Estes e outros dispositivos de entrada podem ser conectados à unidade de processamento 22 através de uma interface de porta serial 52 que é acoplada ao barramento de sistema 26, ou podem ser coletados por outras interfaces, tal como uma interface de porta paralela 54, uma porta de jogos ou um barramento serial universal (USB). Ainda, uma informação pode ser impressa usando-se a impressora 56. A impressora 56 e outros dispositivos de entrada/saída paralelos podem ser conectados à unidade de processamento 22 através da interface de porta paralela 54. Um monitor 58 ou outro tipicamente de dispositivo de exibição também pode ser conectado ao barramento de sistema 26 através de uma interface, tal com uma entrada/saída de vídeo 60. Além do monitor, o ambiente de computação 20 pode incluir outros dispositivos de saída periféricos (não mostrados), tais como alto-falantes ou uma outra saída audível.

O ambiente de computação 20 pode se comunicar com outros dispositivos eletrônicos, tal como um computador, um telefone (com fio ou

sem fio), um assistente digital pessoal, uma televisão ou similar. Para comunicação, o ambiente de computação 20 pode operar em um ambiente em rede usando conexões com um ou mais dispositivos eletrônicos. A Figura 28 descreve o ambiente de computação ligado em rede com um computador remoto 62. O computador remoto 62 pode ser um outro ambiente de computação, tal como um servidor, um roteador, um PC em rede, um dispositivo de par ou outro nó de rede comum, e pode incluir muitos ou todos os elementos descritos acima em relação ao ambiente de computação 20. As conexões lógicas descritas na figura 28 incluem uma rede de área local (LAN) 64 e uma rede de área ampla (WAN) 66. Esses ambientes de rede são lugar comum em escritórios, redes de computador por toda a empresa, intranets e a Internet.

Quando usado em um ambiente de rede de LAN, o ambiente de computação 20 pode ser conectado à LAN 64 através de uma I/O de rede 68. Quando usado em um ambiente de rede de WAN, o ambiente de computação 20 pode incluir um modem 70 ou um outro meio para o estabelecimento de comunicações pela WAN 66. O modem 70, o qual pode ser interno ou externo ao ambiente de computação 20, é conectado ao barramento de sistema 26 através da interface de porta serial 52. Em um ambiente em rede, módulos de programa descritos em relação ao ambiente de computação 20 ou porções dos mesmos podem ser armazenados em um dispositivo de armazenamento de memória remoto residente em ou acessível para um computador remoto 62. Mais ainda, outros dados relevantes para o método implementado em computador para a geração de arquivos executáveis em computador inter-relacionados os quais podem ser produtos de seguros ou estar relacionados a produtos de seguros, e/ou relacionados a aplicação(ões) de gerenciamento de reivindicação de seguro (descrito em maiores detalhes acima) e/ou relacionados à geração de documentos de seguros, e ou relacionados à geração de outros produtos, tais como produtos de imóveis, produtos de serviços financeiros e/ou documentos, podem ser residentes em ou acessíveis através do computador remoto 62. Os dados podem ser armazenados, por exemplo, em um banco de dados de objeto ou em um

relacional. Será apreciado que as conexões de rede mostradas são de exemplo e outros meios de estabelecimento de um enlace de comunicações entre os dispositivos eletrônicos podem ser usados.

- 5 O sistema de computação descrito acima é apenas um exemplo do tipo de sistema de computação que pode ser usado para a implementação do método implementado em computador descrito acima.

Listagem de Números de Referência

	1	GUI
	2a	campo
10	2b	campo
	4a	campo
	6a	campo
	6b	campo
	8	campo
15	10	janela
	12	caixa
	16	botão
	18	botão
	20	ambiente de computação convencional
20	22	unidade de processamento
	24	memória de sistema
	26	barramento de sistema
	28	memória de acesso randômico (RAM)
	30	memória apenas de leitura (ROM)
25	32	drive de disco rígido
	34	drive de disco externo
	36	disco removível
	38	interface de drive de disco rígido
	40	interface de drive de disco externo
30	44	um ou mais programas aplicativos
	46	dados de programa
	48	teclado

	50	mouse
	52	interface de porta serial
	54	interface de porta paralela
	56	impressora
5	58	monitor
	60	entrada/saída de vídeo
	62	computador remoto
	64	rede de área local (LAN)
	66	rede de área ampla (WAN)
10	68	I/O de rede
	70	um modem

REIVINDICAÇÕES

1. Método implementado em computador, para a geração de arquivos inter-relacionados executáveis em computador em um lado de servidor, caracterizado pelo fato de que compreende as etapas de:

- 5 - provisão em um computador de cliente em um lado de cliente de uma interface gráfica de usuário associada a um código legível por máquina em uma linguagem de máquina, onde a interface gráfica de usuário exibe dados substancialmente em texto claro (S1);
- seleção e associação (S2) de duas ou mais seções de código legível por máquina em linguagem de máquina no referido computador de cliente no lado de cliente usando-se a referida interface gráfica de usuário;
- 10 - verificação implementada em computador de uma coerência (S3) do referido código legível por máquina selecionado no dito computador do cliente no lado de cliente antes da compilação do referido código legível por máquina selecionado no lado de cliente, e extração do resultado da referida verificação de coerência substancialmente em texto claro, em que:
- 15 - a verificação da coerência do referido código legível por máquina selecionado compreende
 - a verificação implementada em computador da coerência das
 - 20 referidas seções selecionadas de código legível por máquina e/ou
 - a verificação implementada em computador da coerência das referidas associações das referidas seções selecionadas de código legível por máquina; e
 - mediante a aprovação da coerência no computador do cliente
 - 25 no dito lado do cliente, a compilação (S4, S5) das duas ou mais seções selecionadas e associadas de código legível por máquina em um computador servidor em um lado de servidor para a criação de dois ou mais arquivos executáveis inter-relacionados, os quais são executáveis pelo servidor,
 - sendo que a etapa de verificação implementada em computador
 - 30 (S3) da coerência compreende uma verificação implementada em computador da compatibilidade de uma primeira seção de código usada para a geração de um primeiro arquivo executável com uma segunda seção de código

usada para a geração de um segundo arquivo executável, e

em que a etapa de verificação implementada em computador (S3) da coerência compreende:

- com base no código legível por máquina selecionado, a estimativa no lado de cliente de parâmetros de tempo de execução, antes da compilação das seções selecionadas de código legível por máquina, em que os parâmetros de tempo de execução estão associados aos arquivos executáveis quando da execução dos arquivos executáveis no lado de servidor.

2. Método, de acordo com a reivindicação 1, caracterizado pelo fato que a seleção e a associação (S2) de duas ou mais seções de código legível por máquina em linguagem de máquina compreendem a introdução de regras em texto claro usando a referida interface gráfica de usuário e a provisão de um código legível por máquina em linguagem de máquina equivalente ao referido texto claro usando-se um gerador.

3. Método, de acordo com qualquer uma das reivindicações 1 ou 2, caracterizado pelo fato de que uma ou mais das seções de código legível por máquina são reusadas para compilação (S4, S5) dos dois ou mais arquivos executáveis inter-relacionados.

4. Método, de acordo com qualquer uma das reivindicações 1 a 3, caracterizado pelo fato de que pelo menos um arquivo executável de primeiro nível é compilado, o qual é inter-relacionado com pelo menos um arquivo executável de nível i , em que pelo menos um arquivo executável de nível i é inter-relacionado com pelo menos um arquivo executável de nível $(i+1)$, em que i é um número natural maior do que 1.

5. Método, de acordo com a reivindicação 4, caracterizado pelo fato de que pelo menos um arquivo executável de nível i e pelo menos um arquivo executável de nível $(i+1)$ são compilados a partir de uma ou mais seções de código legível por máquina, em que uma ou mais das seções de código legível por máquina compreendem um código legível por máquina similar, de preferência substancialmente idêntico.

6. Método, de acordo com qualquer uma das reivindicações 1 a 5, caracterizado pelo fato de que compreende a etapa adicional de verifica-

ção (S6) da integridade da execução de todos os arquivos executáveis pela execução de todos os arquivos executáveis no lado de servidor e a comparação da performance no lado de servidor com uma especificação determinada no lado de cliente.

5 7. Método, de acordo com qualquer uma das reivindicações 1 a 6, caracterizado pelo fato de que as etapas no lado de cliente são realizadas em um cliente remoto a um servidor, e as etapas no lado de servidor são realizadas em um servidor remoto ao cliente.

10 8. Sistema baseado em computador para a geração de arquivos executáveis em computador inter-relacionados em um lado de servidor, caracterizado pelo fato de que compreende em um computador de cliente em um lado de cliente:

- uma interface gráfica de usuário em um lado de cliente, onde:
 - a interface gráfica de usuário está associada a um código legível por máquina,
 - a interface gráfica de usuário exibe dados substancialmente em texto claro,
 - a interface gráfica de usuário é adaptada para selecionar duas ou mais seções de código legível por máquina no referido computador de cliente no lado de cliente;
 - um verificador de coerência o qual é adaptado para
 - verificar uma coerência do referido código legível por máquina selecionado no referido computador de cliente no lado de cliente, e
 - extrair o resultado da referida verificação de coerência substancialmente em texto claro; e
 - compreendendo em um computador servidor no lado do servidor:
 - um compilador para compilação das duas ou mais seções selecionadas de código legível por máquina em um lado de servidor para a criação de dois ou mais arquivos executáveis inter-relacionados os quais são executáveis pelo servidor, e
 - compreendendo ainda um estimador, o qual é adaptado para

estimar no lado de cliente com base nos parâmetros de tempo de execução de código legível por máquina selecionados antes da compilação as seções selecionadas de código legível por máquina,

- 5 em que os parâmetros de tempo de execução são associados aos arquivos executáveis quando da execução dos arquivos executáveis no lado de servidor.

9. Sistema baseado em computador, de acordo com a reivindicação 8, caracterizado pelo fato de que a interface gráfica de usuário compreende um gerador o qual é adaptado para prover um código legível por máquina em linguagem de máquina equivalente ao referido texto claro como
- 10 uma entrada através da interface gráfica de usuário.

FIG. 1

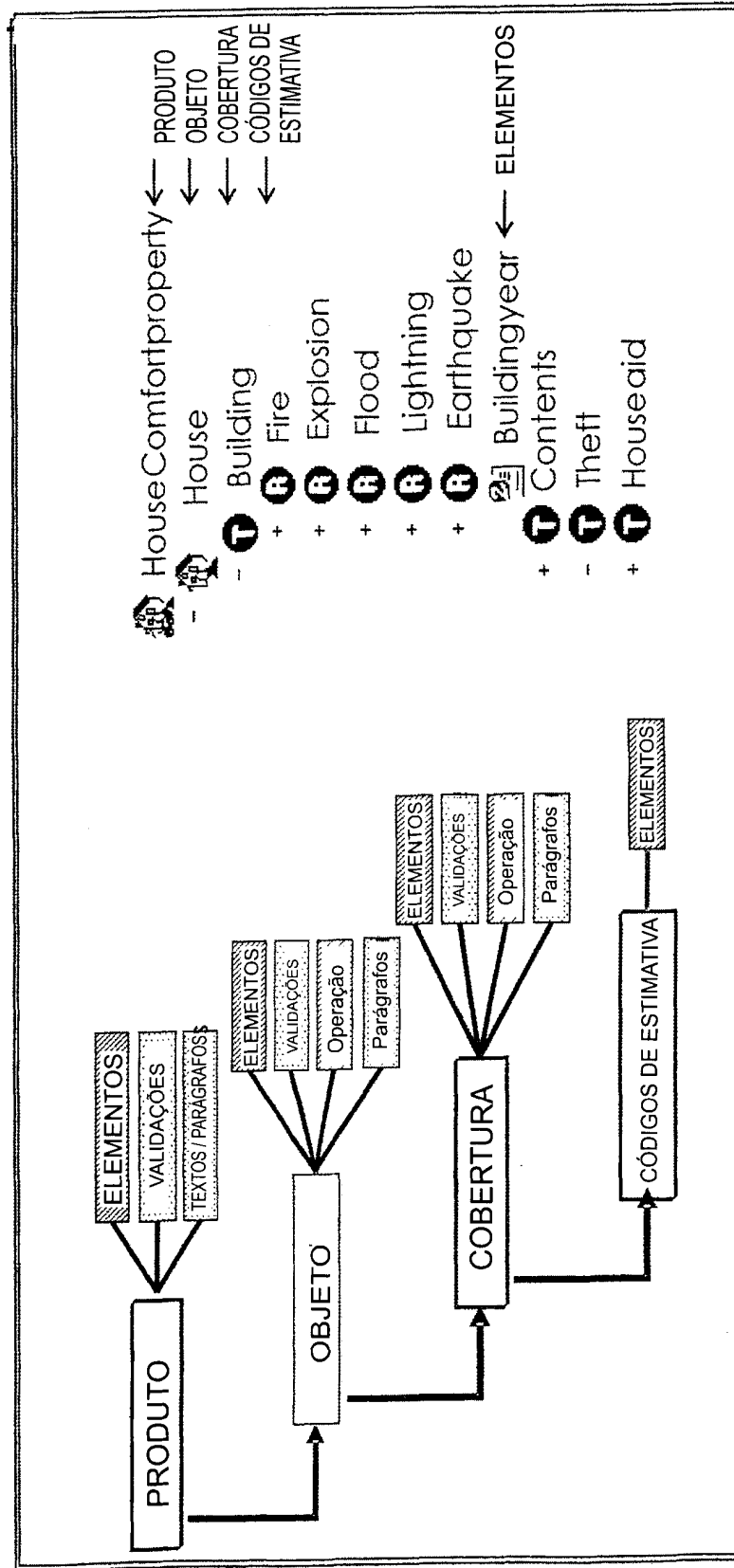


FIG. 2

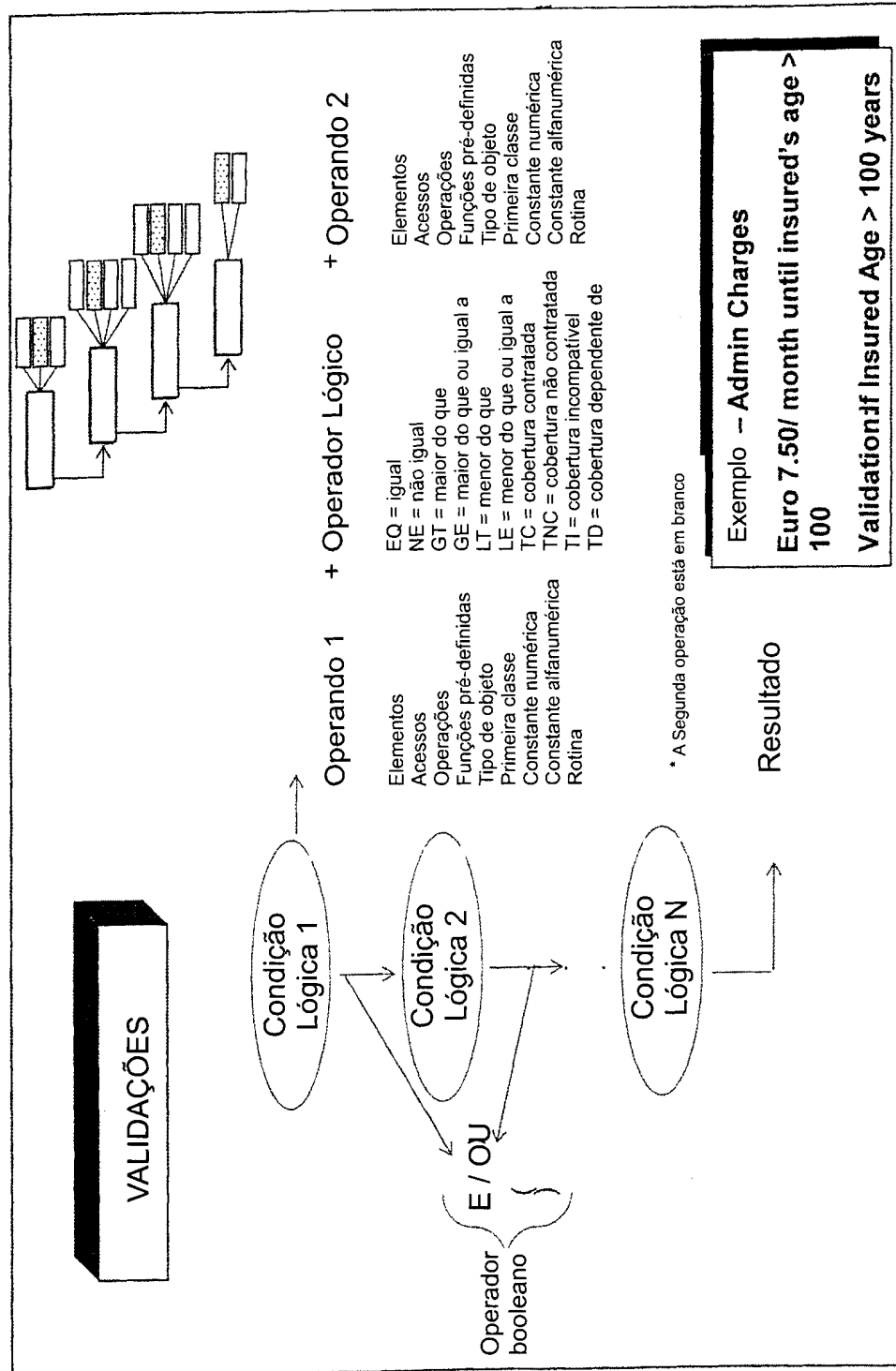
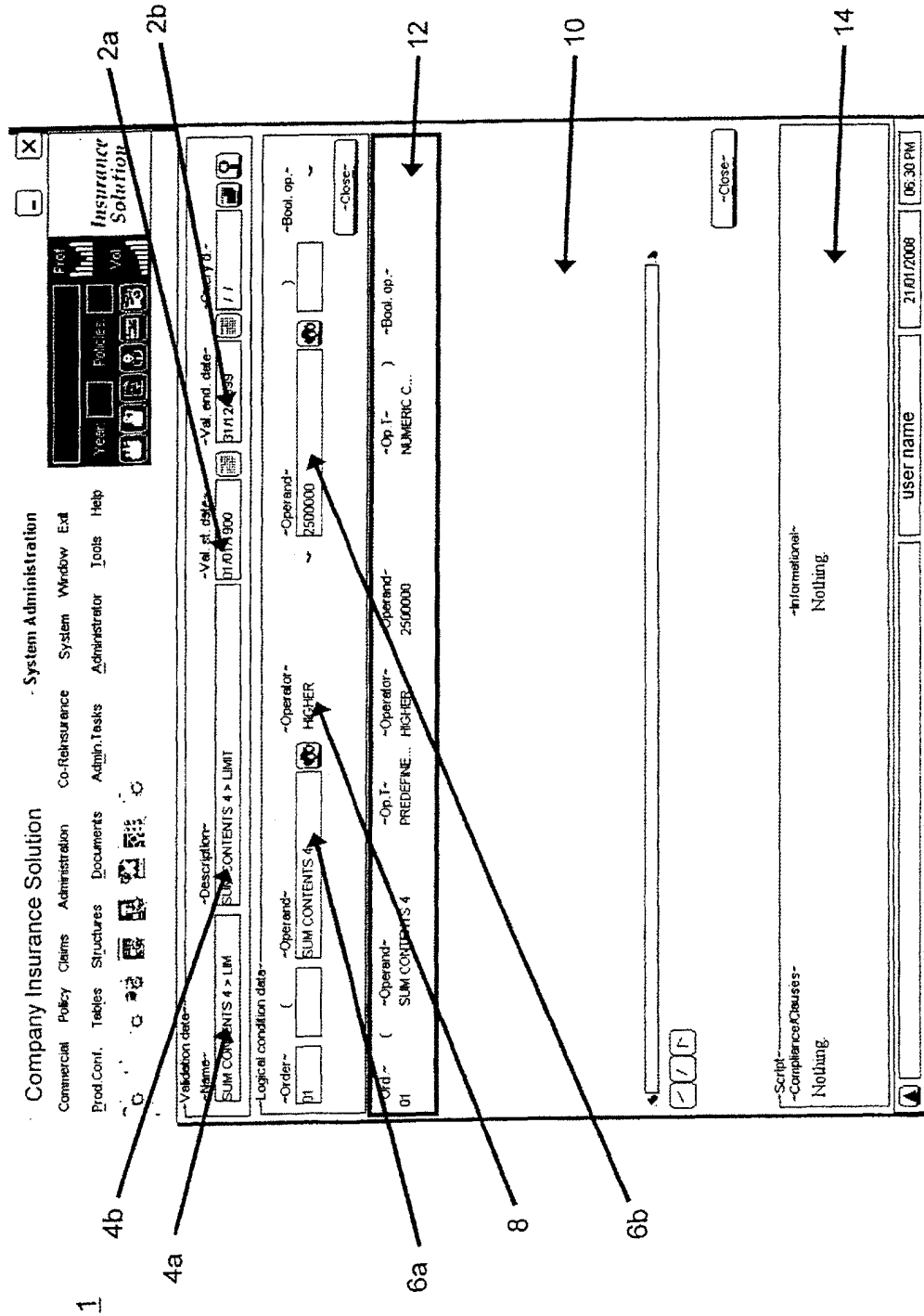


FIG. 3



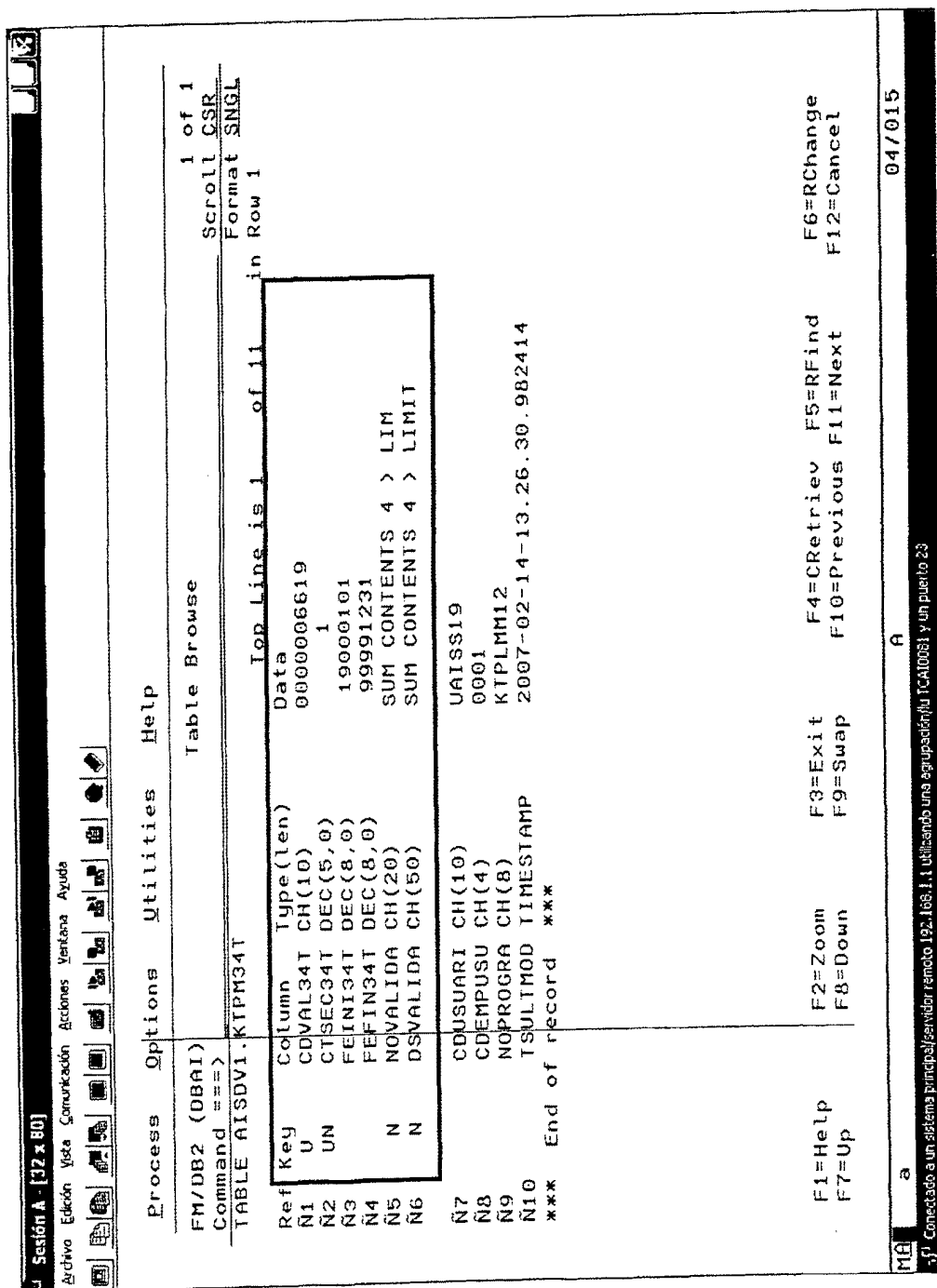


FIG. 5a

Sesión A - [32 x 80]
 Archivo Edición Vista Comunicación Acciones Ventana Ayuda

Process	Options	Utilities	Help
FM/DB2 (DBAI)		Table Browse	
Command ::= > TABLE AISDV1.KIPA78T CDCCP78T CTSEC78T INOCP78T ICTCP78T CDORD78T CQCCCH78T ICTCH78T INMCHIJO N1 N2 N3 N4 N5 N6 N7 N8 CH(10) DEC(5,0) CH(2) CH(2) CH(10) CH(1) CH(2) <UN-+> <U <UN <U <N-+> <N <N 0000006619 1 01 01 0000006178 R 0000006619 1 01 02 N *** - End of data - *** R1 R2			
F1=Help F7=Up	F2=Zoom F8=Down	F3=Exit F9=Swap	F4=CRetriev F5=RFind F10=Left F11=Right F6=RChange F12=Cancel

MA a A ^ 04/015
 Conectado a un sistema principal/servidor remoto 192.168.1.1 utilizando una agrupación/lu TCAD0031 y un puerto 23



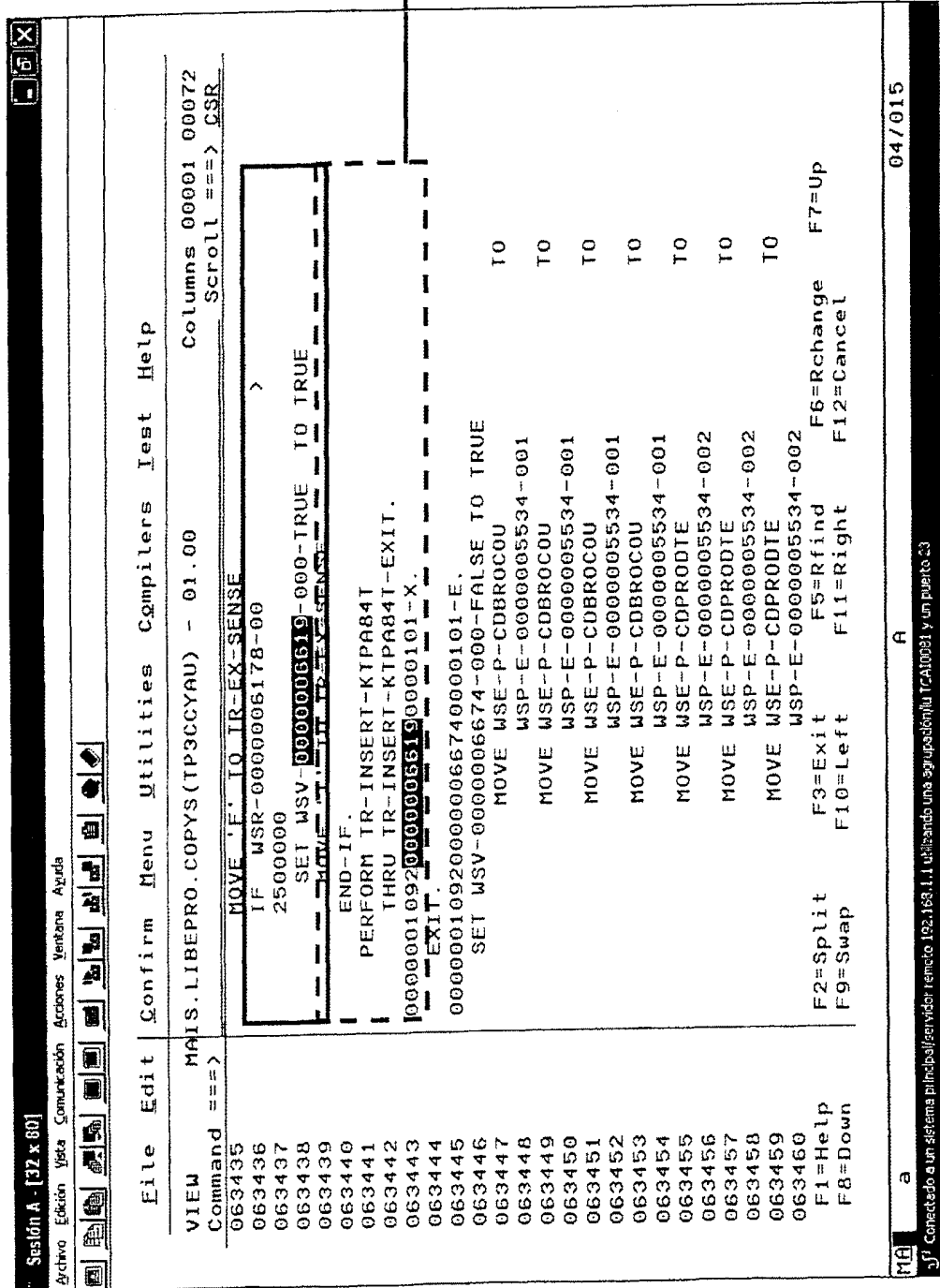


FIG. 7a

Company Insurance Solution

Commercial Policy Clients Administration Co-Renewance System Window Exit

Prod Cont. Tables Structures Documents Admin Tasks Administrator Tools Help

Insurance Solution

Year: Policies: Vol:

-Access definition- -Access query-

-Access data-

-Access name- ACC_SIC_2ND -Access description- Access to SIC table by 2nd Activity Level, as 3rd -Access type- Corporate Table

-Table code- KPTISIT -Table description- SIC table -Non-Data Access- No

-Query by date-

-Val. st. date- 01/01/1900 -Val. end. date- 31/12/9999

-Keys- -Results-

-Key's criteria-

-Key- Beginning date

-Operator- LOWER OR EQUAL

-Operand name / Constant- RATING DATE

-Key-

-Format-	-Operator-	-Operand type-	-Operand-
Beginning date	DEC, 9, 0	LOWER OR EQUAL	RATING DATE
Broker country	CHAR, 3	EQUAL	Broker Country
Activity code 2nd level	CHAR, 8	EQUAL	2. ACTIVITY LEVEL
Activity code 3rd level	CHAR, 8	EQUAL	ALPHANUMERIC CONSTA...

-Script- -Compliance/Clauses-

Nothing

-Informational-

Nothing

-Close-

-Close-

user name 21/01/2008 08:23 PM

FIG. 7b

Company Insurance Solution

Commercial Policy Claims Administration Co-Reinsurance System Window Exit

Prod Conf. Tables Structures Documents Admin Tasks Administrator Tools Help

System Administration

Prof. Vol. Policies: Year: 2008

-Access definition- -Access query-

-Access data-

-Access name- ACC_SIC_2ND

-Access description- Access to SIC table by 2nd Activity Level, as 3rd

-Access Type- Corporate Table

-Table code- CPTST

-Table description- SIC Table

-Non-Data Access- No

-Query by date- / /

-Val. st. date- 01/01/1900

-Val. end date- 31/12/9999

-Keys- **-Rest all-**

-Key's criteria-

-Key- Beginning date

-Operator- LOWER OR EQUAL

-Operand name / Constant- RATING DATE

-Close-

-Key-	-Formal-	-Operator-	-Operand type-	-Operand-
Activity code 3rd Level	CHAR, 8	EQUAL	ALPHANUMERIC CONSTA...	
Corporate table currency code	CHAR, 3	EQUAL	ELEMENT	POLICY CURRENCY
Commercial product code	CHAR, 10	EQUAL	ELEMENT	COMM. PRODUCT
Technical product code	CHAR, 10	EQUAL	ELEMENT	TECH. PRODUCT

-Script-

-Compliance Clauses- Nothing

-Informational- Nothing

-Close-

user name 21/01/2008 08:23 PM

FIG. 8a

Company Insurance Solution

CONTRACTS POLICY CLAIMS ADMINISTRATION

Prod. Cont. Tables Structures Documents Admin. Tasks Administrator Tools Help

Prof. Val. Policies Year: 2003

System Administration

-Access definition- -Access query-

-Access name- ACC_SIC_2ND -Access description- Access to SIC table by 2nd Activity Level, as 3rd -Access Type- Corporate Table

-Table code- KPT1SIT -Table description- SIC table -Non-Data Access- No

-Query by date- -Val. st. date- 01/01/1900 -Val. end date- 01/01/2099

-Keys-	-Results-
-Code- COUSPROD POBULBR POLUPRRW POLUSEXP POLUPRON POLUPROA POLEMTIP POLEMON POLUPUBL	-Name- US product covered Building Base Rate used in Base P Rate % Products liability rest of the Rate % Products liability US exports Rate % Products liability DINC Rate % 1st party recall liability Rate % 1st party Rate % public liability DINC limit Rate % public liability
	-Code- COACTGRU COMBCLAS CORETORN COHCLAS COUSPROD COVYCLAS IMBASLIM IMPROLIM POBULBR
	-Name- Group the 2nd level activity belong MB Class Return code Theft. Class US product covered VV Class Base Capacity limit Protected Capacity limit Building Base Rate used in Base P

-Script-
-Compliance Clauses-
Nothing

-Informational-
Nothing

-Close-

user name 21/01/2008 06:23 PM

FIG. 8b

Company Insurance Solution
 Commercial Policy Claims Administration Co-Reinsurance System Window Exit
 Prof Conf Tables Structures Documents Admin Tasks Administrator Tools Help

Insurance Solution
 Prof Policies Year Policies Vol

Access query
 -Access definition- -Access query-
 -Access data-
 -Access name- ACC_SIC_2ND -Access description- Access to SIC table by 2nd Activity Level, as 3rd -Access Type- Corporate Table
 -Table code- K1PT5IT -Table description- SIC table -Non-Data Access- No
 Query by date- / / -Val. st. date- 01/01/1900 -Val. end. date- 31/12/9999

-Keys-	-Results-
-Code- POLEMP POBUSINT POSTOCER POPRARE PODOACMD POPAACMD POTRACMD COMECLAS	-Name- Rate % acc pollution BI Base Rate Stock base rate Rate % PI Activity Modifier D&O Activity Modifier PA Activity Modifier Crime Activity Modifier Travel MB Class
-Code- POBUSINT POCRACMD PODOACMD POLEMON POLEMP POLEMP POLIPRON POLIPRPA POLIPRRV	-Name- BI Base Rate Activity Modifier Crime Rate % Public liability D&O Rate % Public liability D&O limit Rate % acc pollution Rate % third party Rate % Products liability D&O Rate % 1st party recall liability Rate % Products liability rest of the

-Script-
 -Compliance/Clauses-
 Nothing

-Informational-
 Nothing

-Close-

USER name 21/01/2008 06:24 PM

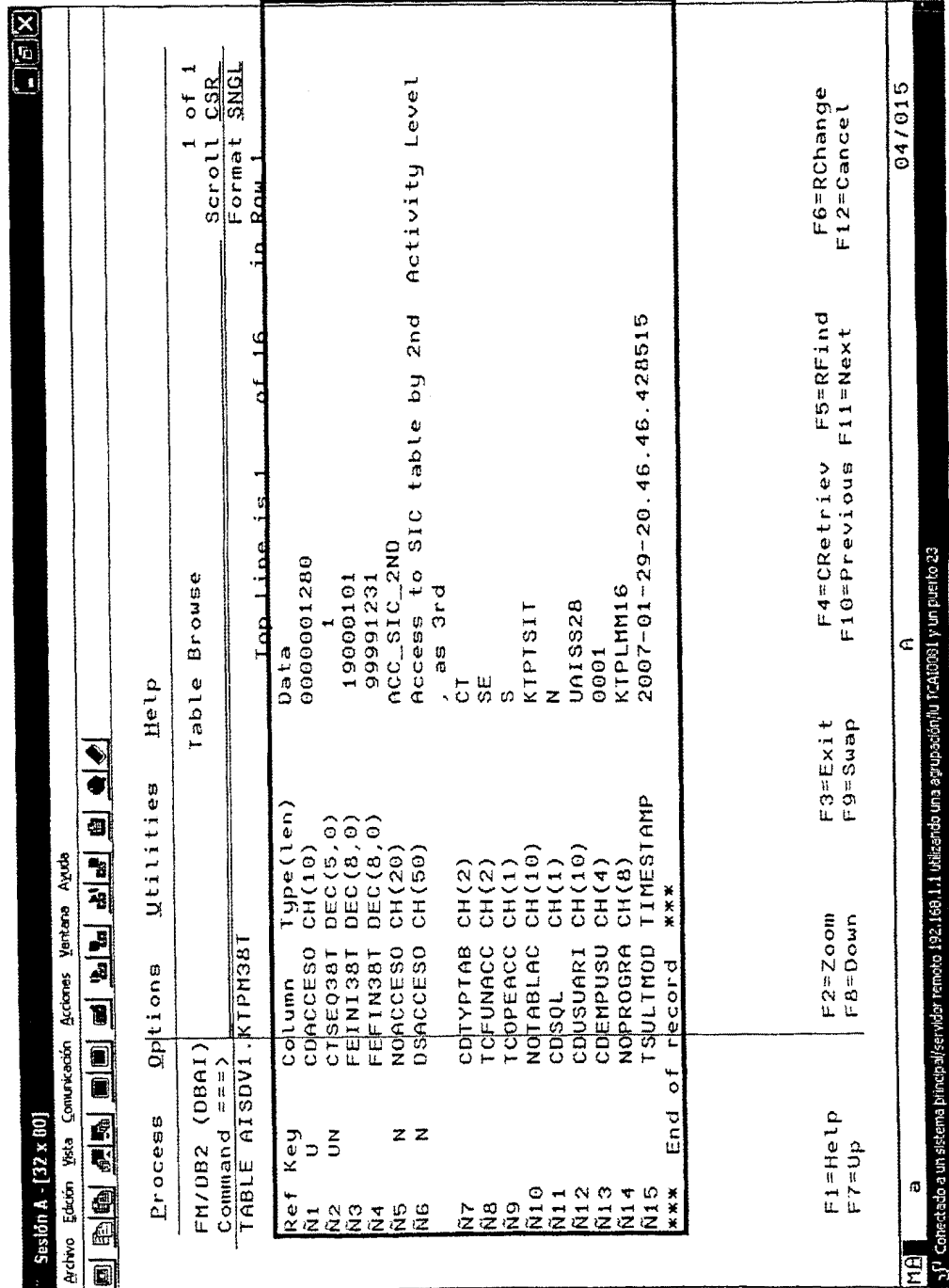


FIG. 10a

Sesión A - [32 x 80]

Archivo

Edición

Vista

Comunicación

Acciones

Ventana

Ayuda

Process

Options

Utilities

Help

Table Browse

1 of 24

Scroll CSR

Format Table

FM/DB2 (DBAI)

Command ==>

TABLE ALSCW1 XIPAS01

CDACCESO CTSEQ38T CDRES30T NORA30T

N1 N2 N3 N4

CH(10) DEC(5,0) CH(8) CHARACTER(50)

<U--+> <U--+> <U--+> <U--+>

1 CDACGRU Group the 2nd level activity belongs to

1 CDMBCLAS MB Class

1 CDRETORN Return code

1 CDTHCLAS Theft Class

1 CDUSPROD US product covered

1 CDVVCLAS VV Class

1 IMBASLIM Base Capacity limit

1 IMPROLIM Protected Capacity Limit

1 POBUILBR Building Base Rate used in Base Perils

1 POBUSINT BI Base Rate

1 POCRACMD Activity Modifier Crime

1 PODOACMD Activity Modifier D&O

1 POLIEMDN Rate % Public liability DINC limit

1 POLIEMPO Rate % acc pollution

1 POLIEMTP Rate % third party

1 POLIPRPN Rate % Products liability DINC

1 POLIPRPA Rate % 1st party recall liability

1 POLIPRRW Rate % Products liability rest of the world

1 POLIPUBL Rate % public liability

1 POLUSEXP Rate % Products US exports

1 POPACEXP Activity Modifier PA

F2=Zoom F3=Exit F4=CRetrieval F5=RFind F6=RCChange

F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel

MA

a

A

04/015

Conectado a un sistema principal/servidor remoto 192.168.1.1 utilizando una agrupación de TCAs0001 y un puerto 23

FIG. 10b

Sesión A - [32 x 80]			
Archivo Edición Vista Comunicación Acciones Ventana Ayuda			
Process	Options	Utilities	Help
FM/DB2 (DBA1)		Table Browse	22 of 24
Command ==>			Scroll CSR
TABLE AISDV1.KTPA301			Format IABL
CDACCESO	CISE0381	CORES301	NORA301
N1	N2 N3	N4	
CH(10)	DEC(5,0)	CH(8)	CHARACTER(50)
<U--+----->	<U--+>	<U--+>	<-----1-----2-----3-----4----->
00000001280	1 POPIRATE	Rate % PI	
00000001280	1 POSTOCBR	Stock base rate	
00000001280	1 POTRACMD	Activity Modifier	Travel
****	End of data	****	
F1=Help F7=Up	F2=Zoom F8=Down	F3=Exit F9=Swap	F4=CRetrie F10=Left
		F5=RRFind F11=Right	F6=RChange F12=Cancel
MA a			04/015

3) Conectado a un sistema principal/servidor remoto 192.168.1.1 utilizando una agrupación/ju TCAID001 y un puerto 23

FIG. 11a

Sesión A - [32 x 80]

Archivo

Edición

Vista

Comunicación

Acciones

Verdadero

Ayuda

Process

Options

Utilities

Help

FM/DB2 (DBAI)

Command ==>

TABLE AISDVI.KTPA37I

CDACCESO CTSEQ38I CDORD37I

N1 N2 N3

CH(10) DEC(5,0) CH(2)

<U---+> <U---+>

**** Top of data ****

00000001280 1 01

00000001280 1 02

00000001280 1 03

00000001280 1 04

00000001280 1 05

00000001280 1 06

00000001280 1 07

**** End of data ****

NOCLAVEA TCOPERCL

N4 N5

CH(8) CH(8)

<---+> <---+>

FEINICIO <=

CDPROCOU =

TCACTIV2 =

TCACTIV3 =

TCCDMONE =

CDPRODCCO =

CDPRODIE =

CDOPERAN

N6

CH(10)

<-N-t-->

FETARIFA

CDPROCOU

TCPACI12

TCCDMONE

CDPRODCCO

CDPRODIE

VALOR37I

N7

CHARACTER(20)

<---+>

0 of 7

Scroll CSR

Format LABEL

F1=Help

F2=Zoom

F3=Exit

F4=CRetrie

F5=RFInd

F6=RChange

F7=Up

F8=Down

F9=Swap

F10=Left

F11=Right

F12=Cancel

MA

a

04/015

3) Conectado a un sistema principal/servidor remoto 192.168.1.1 utilizando una agnucodificación TCAT0031 y un puerto 23

R1
R2
etc



FIG. 12a

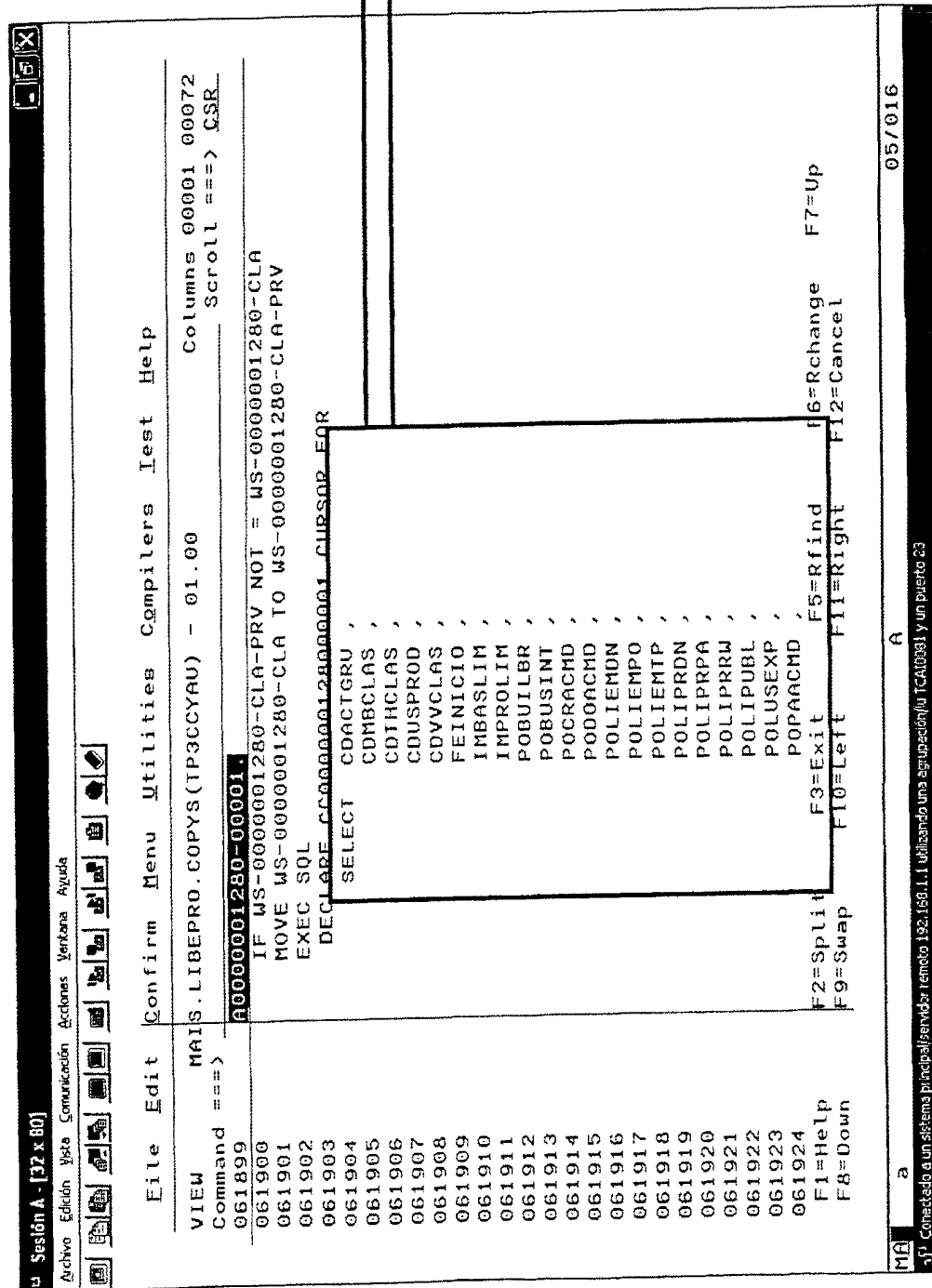




FIG. 12c

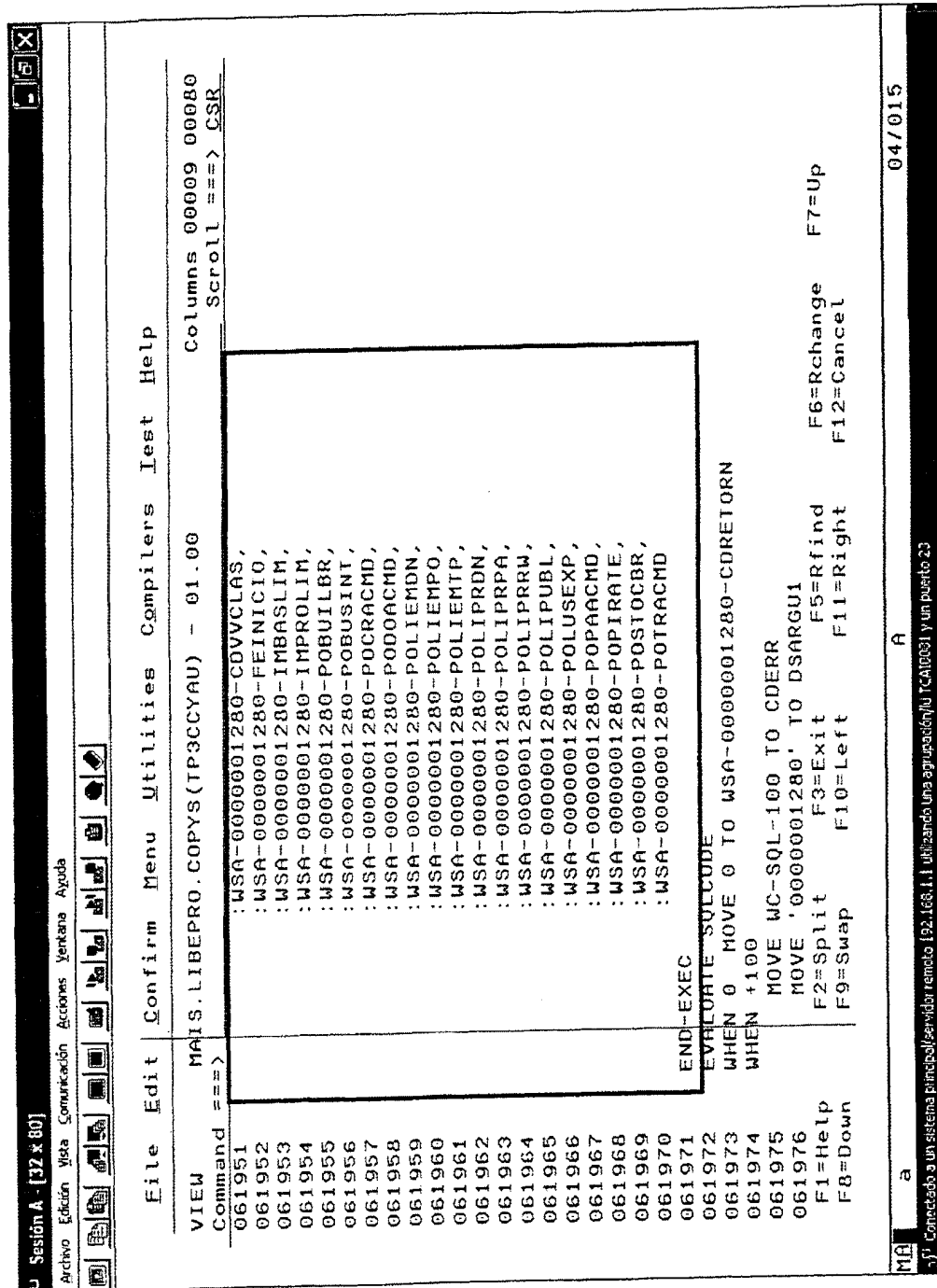


FIG. 12d

Sessão A - [32 x 80]

Arquivo Edição Vista Comunicação Ações Janela Ajuda

File Edit Confirm Menu Utilities Compilers Test Help

VIEW MAIS.LIBEPRO.COPYS(IP3CCYAU) - 01.00 Columns 00001 00072
Command ==> Scroll ==> CSR

```

061977 MOVE '+100 NOT ALLOWED' TO DSARGU2
061978 PERFORM CLOSE-CC0000000128000001
061979 THRU CLOSE-CC0000000128000001-X
061980 PERFORM 9999-ERROR-MANAG
061981 THRU 9999-ERROR-MANAG-EXIT
061982
061983 WHEN OTHER
061984 MOVE WC-SQL-ERROR TO CDERR
061985 MOVE WC-SELECT TO DSARGU1
061986 MOVE 'KTPSIT01280' TO DSARGU2
061987 MOVE SQLCODE TO DSADERR
061988 PERFORM CLOSE-CC0000000128000001
061989 THRU CLOSE-CC0000000128000001-X
061990 PERFORM 9999-ERROR-MANAG THRU 9999-ERROR-MANAG-EXIT
061991 END-EVALUATE
061992 PERFORM CLOSE-CC0000000128000001
061993 THRU CLOSE-CC0000000128000001-X
061994 END-IF.
061995 A0000001280-00001-X.
061996 EXIT.
061997 EXEC SQL CLOSE CC0000000128000001 END-EXEC
061998 IF SQLCODE < 0
061999 PERFORM GENERATE-ERROR THRU GENERATE-ERROR-EXIT
062000 END-IF.
062001 CLOSE-CC0000000128000001-X.
062002 0000001094000000668700001-E.
F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F9=Swap F10=Left F11=Right F12=Cancel
F8=Down

```

Manipulação de erro

Linhas de fim

04/015

MA a

Conectado a um sistema principal/servidor remoto 192.168.1.1 utilizando uma suposição/ju TCA0001 y un puerto 23

FIG. 13

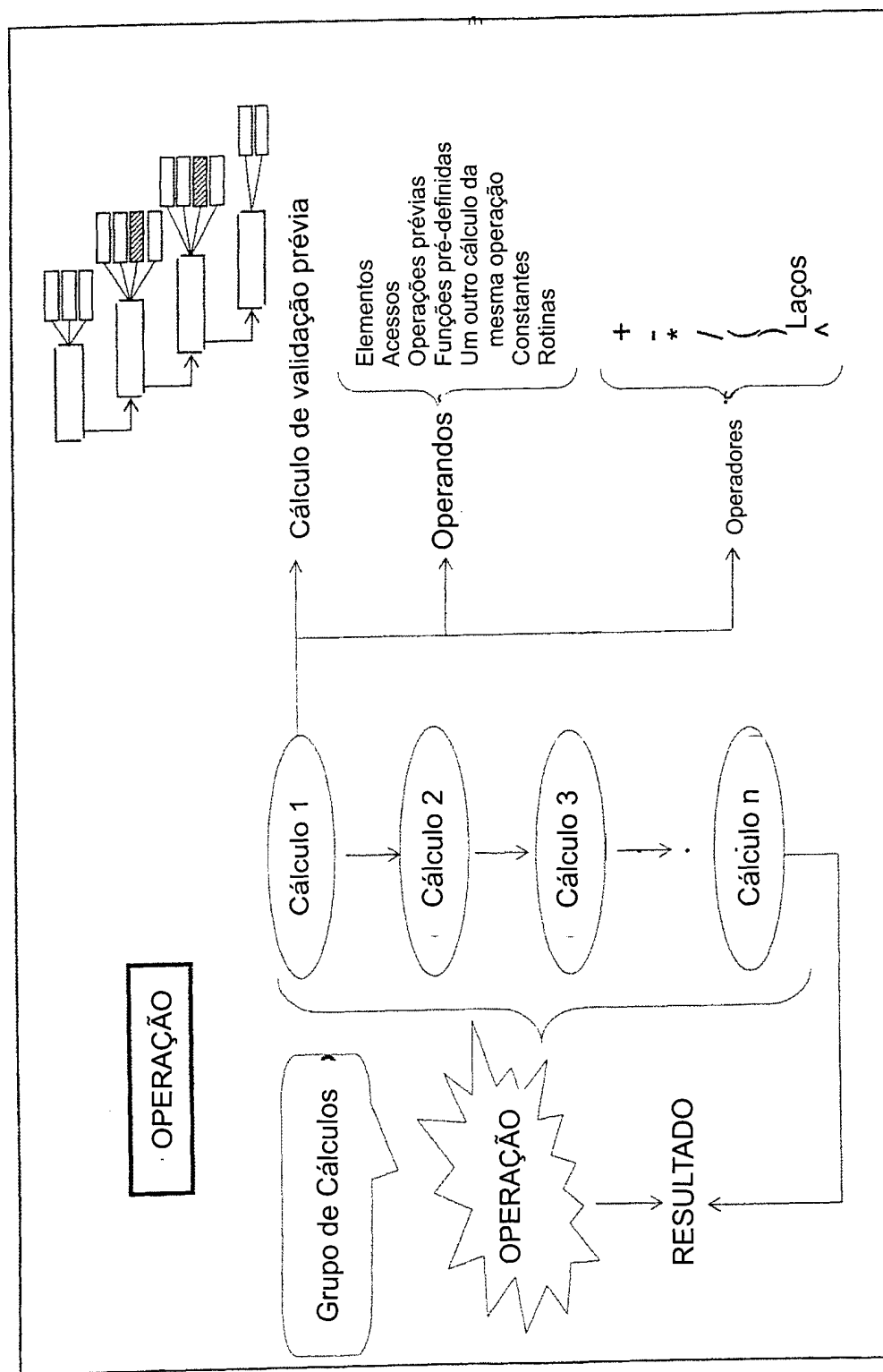


FIG. 14a

Company Insurance Solution

Commercial Policy Claims Administration Co-Reinsurance System Window Exit

Prod Conf. Tables Structures Documents Admin. Tasks Administrator Tools Help

Prof. Vol.

Year: Policies: Vol.

Insurance Solution

Calculations - Operation query -

-Name- CP_DEFAULT_0 -Val. st. date- 01/01/1900 -Val. end. date- 31/12/9999 -Query id- 11

-Data type- INTERMEDIATE VALUE DD: ~ USER -Insurance type- -Pred. func. type-

-Amount-

Calculation data

-Order- 01 -Logical condition- SUM CONTENTS 4 > LIM -Situation- FALSE -Calculation name- C1 -Formal- DEC, 15, 3 -Round- -Truncate-

-Calculation- R:SUM CONTENTS 4] * [0.1] -Opnd- -Operator-

-Formets- -Loop cond-

-Close-

-Order-	-Condition-	-Calculation name-
01	SUM CONTENTS 4 > LIM	C1
05	SUM CONTENTS 4 > LIM	C2
10		C3

-Close-

Script - Compliance/Clauses - Informational - Nothing

-Close-

user name 21/01/2008 06:05 PM

FIG. 14b

Company Insurance Solution

Commercial Policy Claims Administration LO-REINSURANCE System Window Exit

Prod Conf. Tables Structures Documents Admin Tasks Administrator Tools Help

Calculations - Operation query

-Name- -Description- -Val. st. date- -Val. end date- -Query d-

OP_DEFAULT_B OP_DEFAULT_B 01/01/2000 01/12/9999 11

-Data type- -Operation type- -Insurance ty- -Pred. func. type-

INTERMEDIATE VALUE 00: ~ USER ~ -Amount- ~

Calculation data

-Order- -Logical condition- -Situation- -Calculation name- -Format- -Round- -Operator-

05 SUM(Contents 4 > LIM) TRUE ~ C2 DEC, 15, 3 ~ ~

Calculation: 2500000

-Formulas... -Loop cond... -Close-

-Order-	-Condition-	-Situation-	-Calculation name-
01	SUM(Contents 4 > LIM)	FALSE	C1
05	SUM(Contents 4 > LIM)	TRUE	C2
10			C3

-Close-

Script

-Compliance/Clauses- -Informational- -Close-

Nothing Nothing

user name 21/01/2008 06:05 PM

FIG. 14C

Company Insurance Solution

Commercial Policy Claims Administration Co-tenure Insurance System Window Exit

Prod Conf. Tables Structures Documents Admin Tasks Administrator Tools Help

Insurance Solution

Print Policies Year Policies Vol

Calculations - Operation query

Name: OP_DEFAULT_B Description: OP_DEFAULT_B Val. st. date: 01/01/1900 Val. end. date: 31/12/9999 Query d.: / /

Data type: INTERMEDIATE VALUE 00: USER Insurance ty.: -Pred. func. type: -Amount- -Round- -Truncate-

Calculation data

Order: 10 Logical condition: Situation: TRUE Calculation name: DEC, 15, 3 Format: -Opnd... -Operator-

Calculation: C:C1) * (C:C2)

Format **Loop cond.** **Close**

Order	Condition	Situation	Calculation name
01	SUM CONTENTS 4 > LIM	FALSE	C1
05	SUM CONTENTS 4 > LIM	TRUE	C2
10			C3

Script **Close**

Compliance/Clauses: Nothing Informational: Nothing

user name 21/01/2008 06:05 PM

FIG. 15

Sesión A - [32 x 80]

Archivo Edición Vista Comunicación Acciones Ventana Ayuda

Process Options Utilities Help

FM/DB2 (DBAI)

Command ==>

TABLE AISDV1.KIPM56T

Table Browse

1 of 1

Scroll CSR

Format SNGL

Top line is 1 of 16 in Row 1

Ref Key	Column	Type(len)	Data
N1 U	COOPESGT	CH(10)	0000006092
N2 UN	CTSEC56T	DEC(5,0)	1
N3	FEIN56T	DEC(8,0)	19000101
N4	FEFIN56T	DEC(8,0)	99991231
N5 N	NOOPERAC	CH(20)	OP_DEFAULT_8
N6 N	DSOPERAC	CH(50)	OP_DEFAULT_8
N7	INIMPOR	CH(1)	N
N8	TCOPERAC	CH(1)	U
N9	TCIPSTD	CH(1)	
N10	TCIPFUN	CH(1)	
N11	CDELEM	CH(8)	VAINT007
N12	CDUSUARI	CH(10)	UAISS19
N13	CDEMPUSU	CH(4)	0001
N14	NOPROGRA	CH(8)	KIPLMM21
N15	TSULTMOD	TIMESTAMP	2007-02-16-13.02.22.027116
***	End of record	***	

F1=Help F2=Zoom F3=Exit F4=CRetrieV F5=RFInd F6=RChange
F7=Up F8=Down F9=Swap F10=Previous F11=Next F12=Cancel

MA a A 03/015

Conectado a un sistema principal/servidor remoto 192.168.1.1 utilizando una agrupación TCAC0001 y un puerto 23

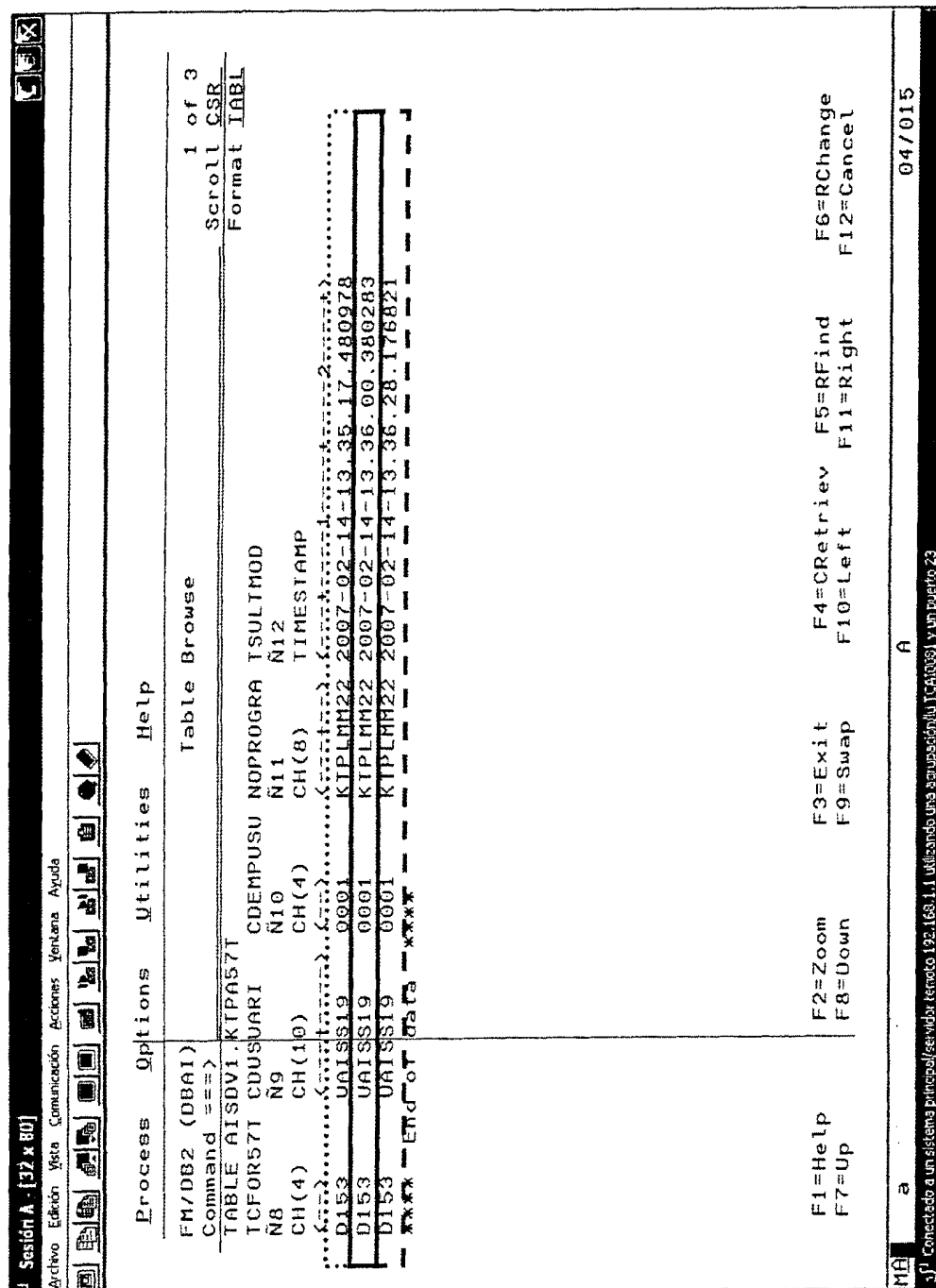


FIG. 17a

Sesión A - [32 x 80]

Archivo Edición Visto Comunicación Acciones Ventana Ayuda

Process Options Utilities Help

Table Browse

0 of 5

Scroll CSR

Format IABL

TABLE AISDV1.KIPA781

CUCCP781 CISEC781 INOCP781 ICTCP781 COORD781 CDCCH781 ICTCH781 INMCH781

N1 CH(10) DEC(5,0) CH(2) N4 CH(2) N5 CH(2) N6 CH(10) N7 CH(1) N8 CH(2)

<UN-+----> <U-++> <UN> <U> <N-+----> - N <N>

**** Top of data ****

00000006092 1 01 CC 01 00000006178 R

00000006092 1 01 CC 02 01 N

00000006092 1 05 CC 01 N

00000006092 1 10 CC 01 C

00000006092 1 10 CC 02 C

**** End of data ****

F1=Help F2=Zoom F3=Exit F4=CRetriev F5=RFind F6=RChange

F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel

MA a R 04/015

Conectado a un sistema principal/servidor remoto 192.168.1.1 utilizando una agrupación de TCA0001 y un puerto 23

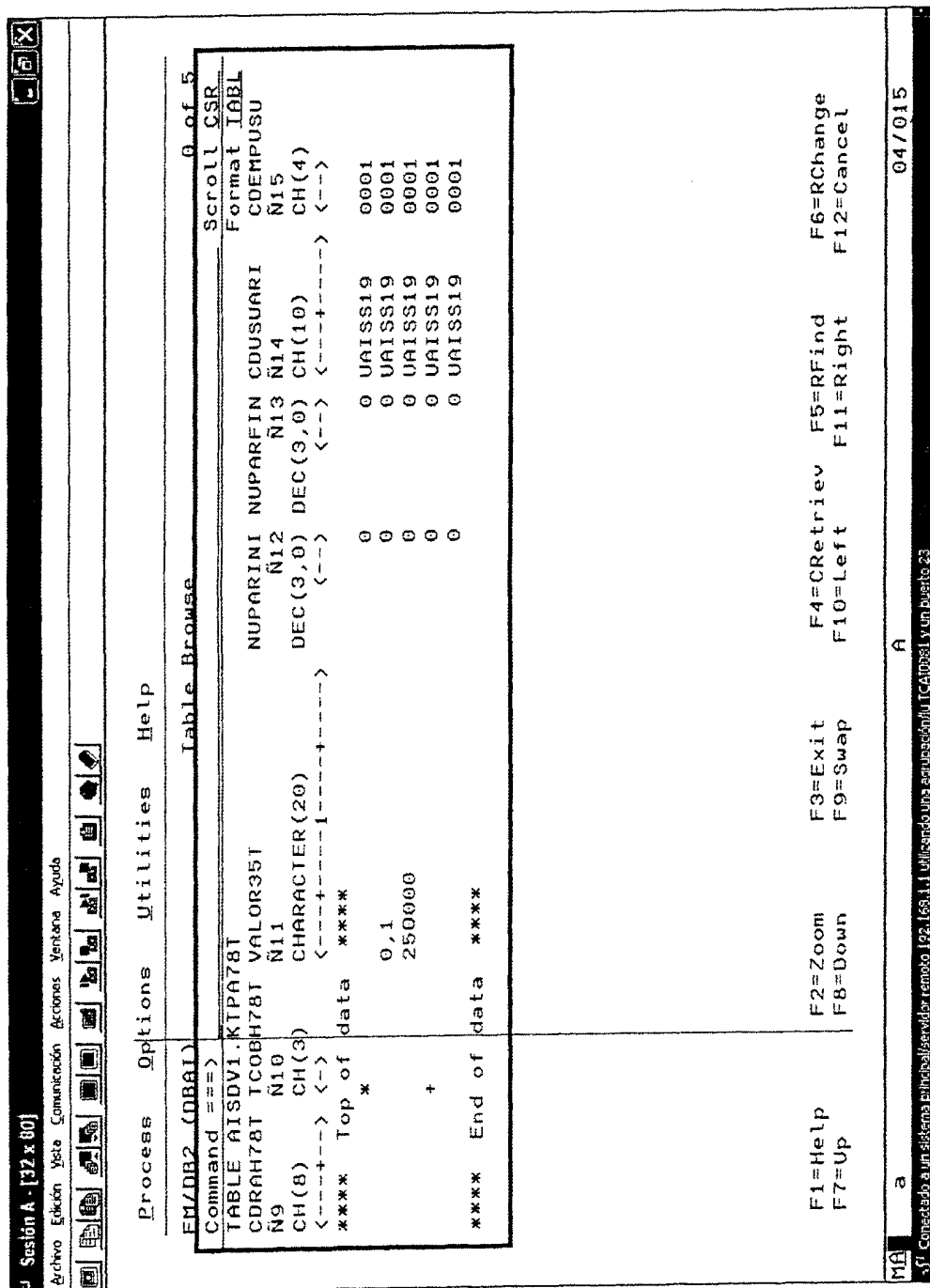


FIG. 18a

Sesión A - [32 x 80]

Archivo Edición Vista Comunicación Acciones Ventana Ayuda

File Edit Confirm Menu Utilities Compilers Test Help

VIEW MAIS.LIBEPRO.COPYS(TP3CCYAU) - 01.00 Columns 00001 00072
Command ==> Scroll ==> CSR

00000001092-00000006092-00001.
009189 PERFORM 00000001092000000006619-E THRU 00000001092000000006619-X
009190 IF WSV-00000006619-000 = 'F'
009191 PERFORM 0000000109200000060920000101
009192 THRU 0000000109200000060920000101-X
009193
009194 END-IF
009195 PERFORM 000000010920000006619-E THRU 00000001092000000006619-X
009196 IF WSV-00000006619-000 = 'V'
009197 PERFORM 0000000109200000060920000105
009198 THRU 0000000109200000060920000105-X
009199 END-IF
009200 PERFORM 0000000109200000060920000110
009201 THRU 0000000109200000060920000110-X
009202 INITIALIZE TR-EX-VALUES
009203 REPLACING ALPHANUMERIC DATA BY SPACES
009204 NUMERIC DATA BY ZERO
009205 MOVE 'OPE' TO TR-EX-TYPE OF TR-EX-VALUES
009206 MOVE '00000006092' TO TR-EX-OPERATION OF TR-EX-LEVEL
009207 MOVE 'N' TO TR-EX-ROUNDING OF TR-EX-OPERATION
009208 MOVE WSC-00000001092-00000006092-10
009209 TO WSO-00000001092-00000006092
009210 MOVE WSO-00000001092-00000006092
009211 TO TR-EX-RESULT OF TR-EX-OPERATION
009212 PERFORM TR-INSERT-KIPA84T
009213 THRU TR-INSERT-KIPA84T-EXIT.
009214 00000001092-00000006092-00001-X.
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

05/028

3) Conectado a un sistema principal/servidor: emodo 192.168.1.1 utilizando una aguja de conexión TCA0099 y un puerto 23

FIG. 18b

Sesión A - [32 x 80]
 Archivo Edición Vista Comunicación Acciones Ventana Ayuda

File	Edit	Confirm	Menu	Utilities	Compilers	Test	Help
VIEW	MAIS.LIBEPRO.COPYS(IP3CCYAU) - 01.00	Columns 00001 00072					Scroll ==> CSR
Command ==>							
032612	COMPUTE WSC-0000001092-	NUMERIC	DATA BY ZERO				
032613	WSR-0000006178-00						
032614	0,1						
032615	ON SIZE ERROR						
032616	PERFORM CALCULT-ERROR THRU CALCULT-ERROR-EXIT						
032617	END-COMPUTE.						
032618	MOVE 'CCL' TO TR-EX-TYPE OF TR-EX-VALUES						
032619	PERFORM TR00001092-00006092-0100001						
032620	THRU TR00001092-00006092-0100001-E.						
032621	00000001092000000060920000101-X.						
032622	EXIT.						
032623	00000001092000000060920000105.						
032624	INITIALIZE TR-EX-VALUES						
032625	REPLACING ALPHANUMERIC DATA BY SPACES						
032626	NUMERIC						
032627	DATA BY ZERO						
032628	COMPUTE WSC-0000001092-0000006092-05						
032629	250000						
032630	ON SIZE ERROR						
032631	PERFORM CALCULT-ERROR THRU CALCULT-ERROR-EXIT						
032632	END-COMPUTE.						
032633	MOVE 'CCL' TO TR-EX-TYPE OF TR-EX-VALUES						
032634	PERFORM TR00001092-00006092-0500001						
032635	THRU TR00001092-00006092-0500001-E.						
032636	00000001092000000060920000105-X.						
032637	EXIT.						
F1=Help	F2=Split	F3=Exit	F5=Rfind	F6=Rchange	F7=Up		
F8=Down	F9=Swap	F10=Left	F11=Right	F12=Cancel			

ME a 04/015

S¹ Conectado a un sistema principal/servidor remoto 192.168.1.1 utilizando una agrupación/lu TCAT0031 y un puerto 23

FIG. 18c

Sesión A - [32 x 80]
Archivo Edición Vista Comunicación Acciones Ventana Ayuda

File Edit Confirm Menu Utilities Compilers Test Help

```

VIEW      MAIS.LIBPRO.COPYS(TP3CCYAU) - 01.00      Columns 00001 00072
Command ==>
032636    00000001092000000060920000105-X.
032637    EXIT.
032638    00000001092000000060920000110
032639    INITIALIZE TR-EX-VALUES
032640    REPLACING ALPHANUMERIC DATA BY SPACES
032641    NUMERIC DATA BY ZERO
032642    COMPUTE WSC-00000001092-0000006092-10 =
032643    WSC-00000001092-00000006092-01 +
032644    WSC-00000001092-0000006092-05
032645    ON SIZE ERROR
032646    PERFORM CALCULT-ERROR THRU CALCULT-ERROR-EXIT
032647    END-COMPUTE.
032648    MOVE 'CCL' TO TR-EX-TYPE OF TR-EX-VALUES
032649    PERFORM TR00001092-00006092-1000001
032650    THRU TR00001092-00006092-1000001-E.
032651    00000001092000000060920000110-X.
032652    EXIT.
032653    000000109200000006674-E.
032654    MOVE '0000006674' TO WF-VALID.
032655    MOVE '000001092000000667400' TO WS-LABEL-INC
032656    MOVE '-' TO WS-LABEL-INC(21:1)
032657    MOVE WS-TRANSACTION-CURRENT TO WS-LABEL-INC(22:1)
032658    PERFORM 00000010920000000667400001-E
032659    THRU 00000010920000000667400001-X.
032660    0000000109200000006674-X
032661    EXIT.
F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F9=Swap      F10=Left      F11=Right      F12=Cancel
F8=Down

```

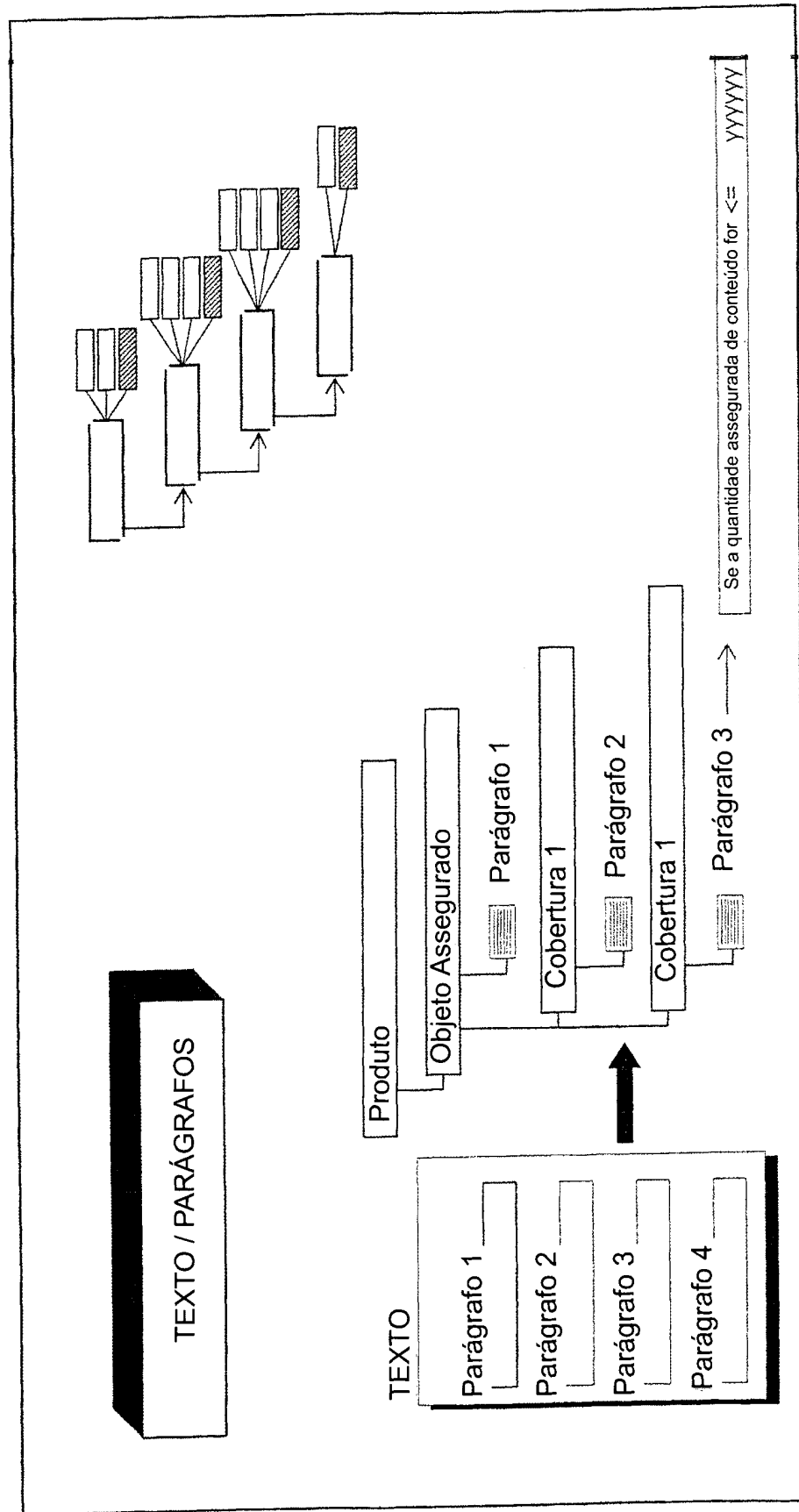
07/035

a

Conectado a un sistema principal/servidor remoto 192.168.1.1 utilizando una agrupación de TCAL0001 y un puerto 23

Salida, Eol

FIG. 19



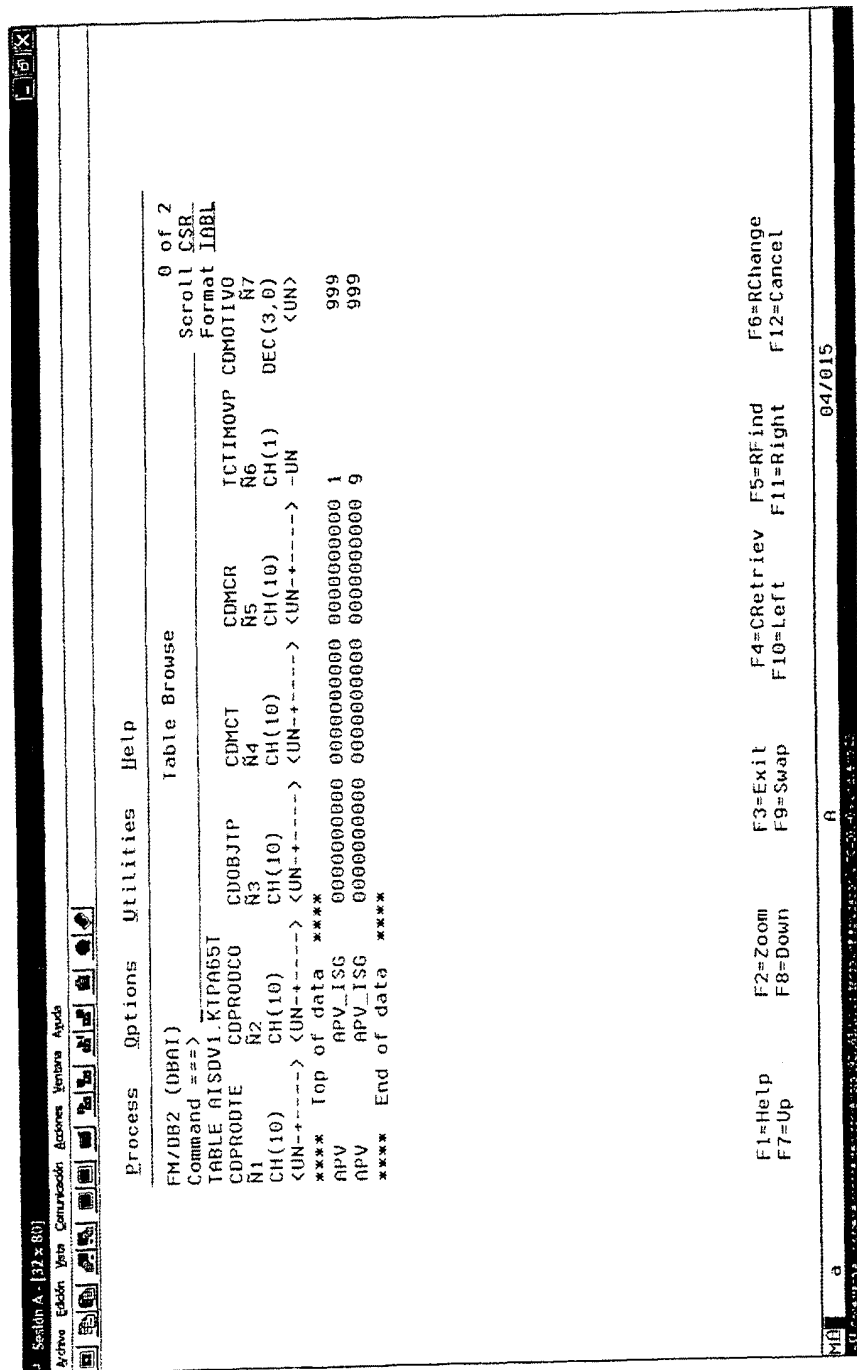


FIG. 20c

Session A - [32 x 80]
 Archivo Edición Vistas Configuración Ayudas Ventana Ayuda

Process Options Utilities Help

Table Browse		0 of 2
FM/DB2 (DBA1)		Scroll CSR
Command ==>		Format LABEL
TABLE AISUV1.KIPA65I		ISULTMOD
TCROLDDES INOBLIGA		N23
N16 N17		CH(8)
CH(3) CH(1)		TIMESTAMP
<->		<-----I-
**** top of data ****		
19000101	99991231 0001	KTPLMM31 2006-09-12-
19000101	99991231 0001	KTPLMM31 2006-09-12-
**** End of data ****		

F1=Help F2=Zoom F3=Exit F4=CRetrieval F5=RFind F6=RChange
 F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel

84/015

FIG. 21a

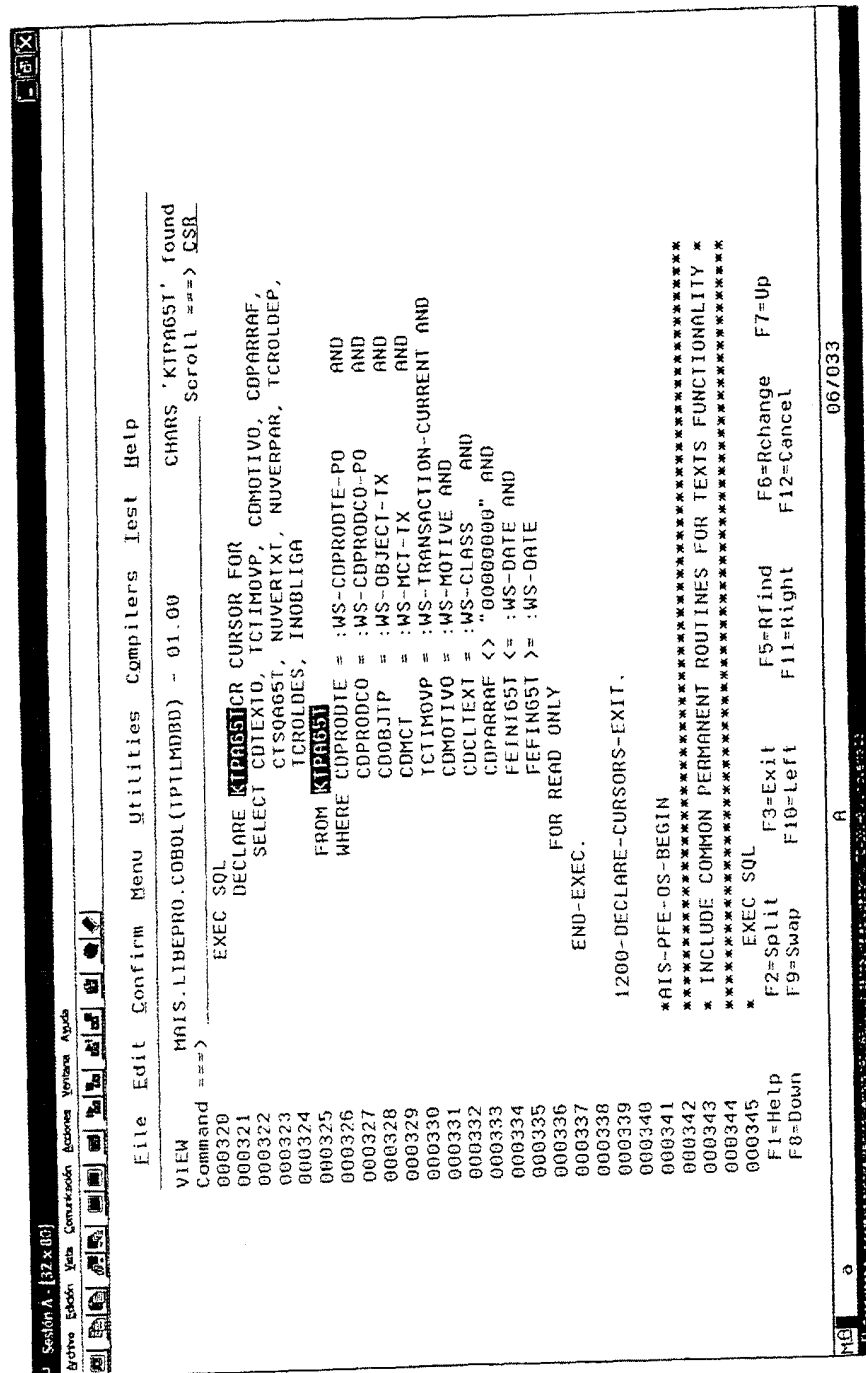


FIG. 21b

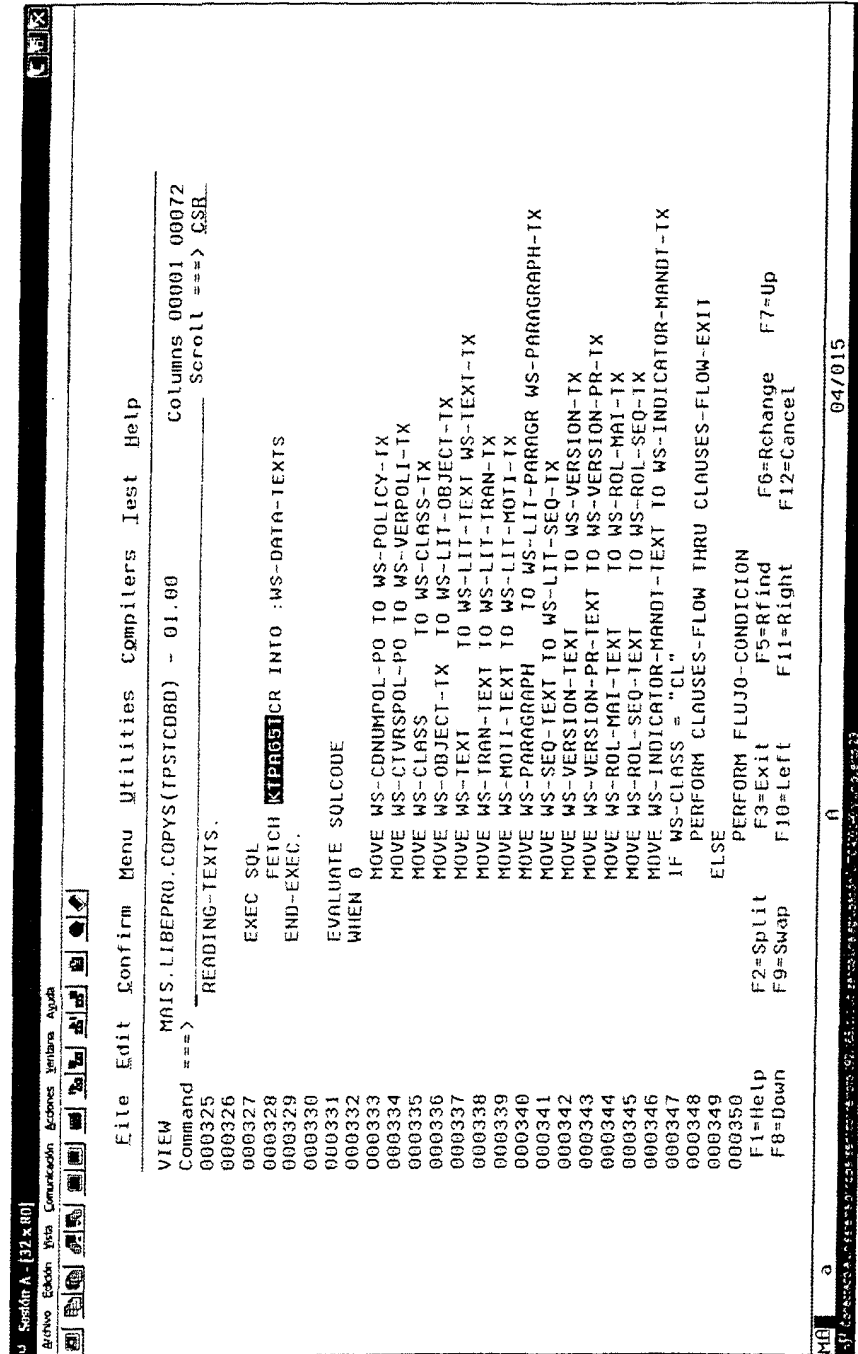


FIG. 22

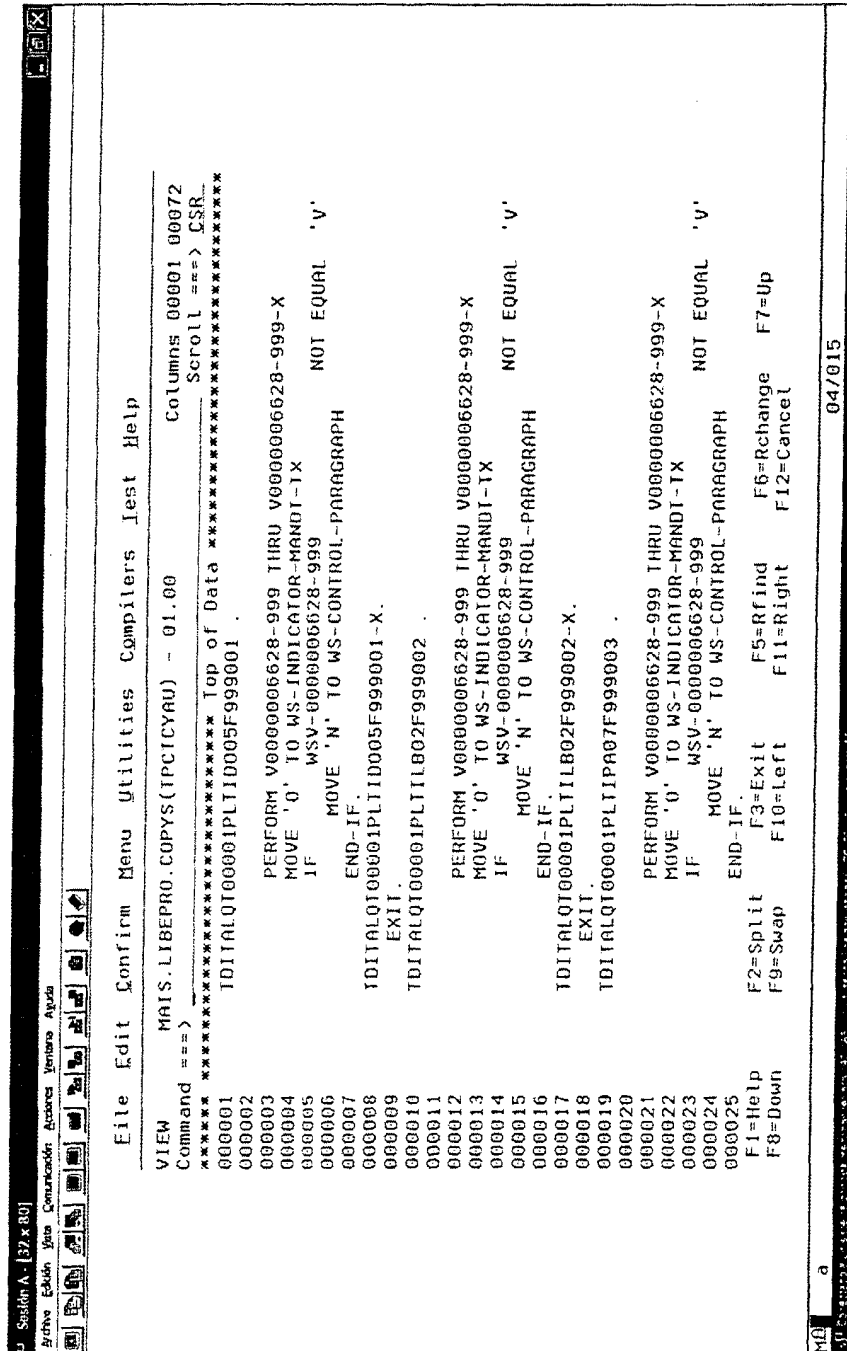


FIG. 23

Ação	VALIDAR	TAXA
------	---------	------

Nível de Administração de Política		
POLÍTICA	OBJETO	COBERTURAS

Chamada de programa		
LER DADOS DE POLÍTICA	LER DADOS DE POLÍTICA	LER DADOS DE POLÍTICA
VALIDAÇÕES DE POLÍTICA	LER DADOS DE OBJETO	LER DADOS DE OBJETO
	VALIDAÇÕES DE OBJETO	LER DADOS DE COBERTURA
		VALIDAÇÕES DE COBERTURA
		TAXA
		ESCREVER RESULTADOS ECONÔMICOS

FIG. 24

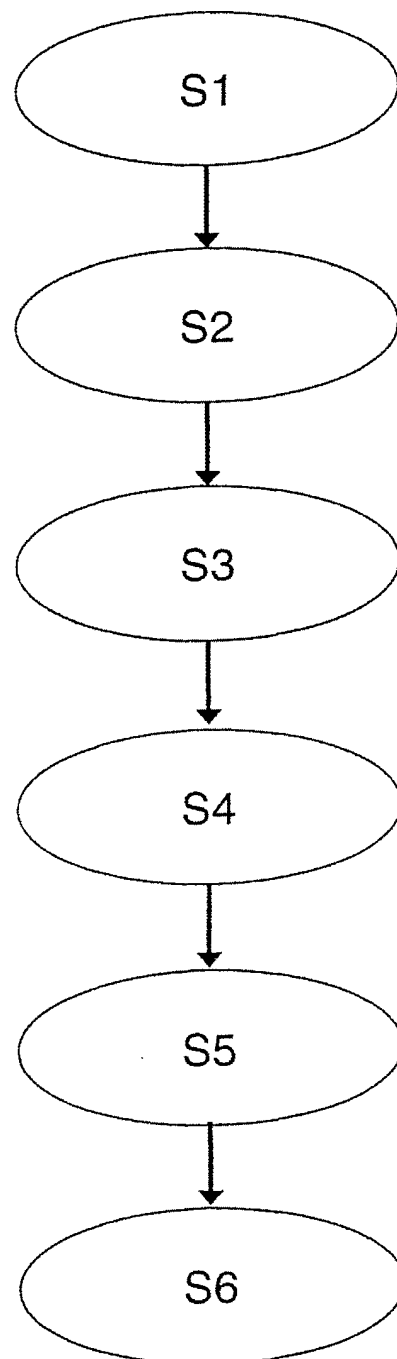


FIG. 25

LOCAL Development, MASTER Environment · **Central Administration**

Party & Agent Policy Admin Customer Accounting Central Admin Window Exit
 Products User conf Tables Business Docs Test & Notes Menus Errors System adm Tools Help

Central Administration / Products / Product design

Product search criteria

Template: Street: Product Name: Description: Type:
 Structure: Description: Relationship type: Status: Insurance type:

Template	Street	Product Name	Description	Type	Status	Ins. type
0	0000000000	0	0	TECHNICAL	DEVELOP...	ACCIDENTS ...
0000JR	0000000000	PRUEBA	PRUEBA	TECHNICAL	DEVELOP...	ACCIDENTS ...
0000JR2	0000000000	PRUEBA2	P2	TECHNICAL	DEVELOP...	AUTOS ...
0000JR	0000000000	PRUEBA	PRUEBA	TECHNICAL	DEVELOP...	AUTOS ...
CORE MOTOR	0000000000	CORE MOTOR	CORE MOTOR	TECHNICAL	DEVELOP...	AUTOS ...
MOTORFLEET	0000000000	MOTOR FLEET	MOTOR FLEET	TECHNICAL	ACTIVE	AUTOS ...
MOTORFLEET	0000000000	STREET	MOTOR FLE	TECHNICAL	ACTIVE	AUTOS ...
MOTORFLEET	0000000000	YACHT PROTECTOR	YACHT PROTECTOR	TECHNICAL	DEVELOP...	PENSIONS ...

16

user name 23/02/2010 02:12 PM

FIG. 26

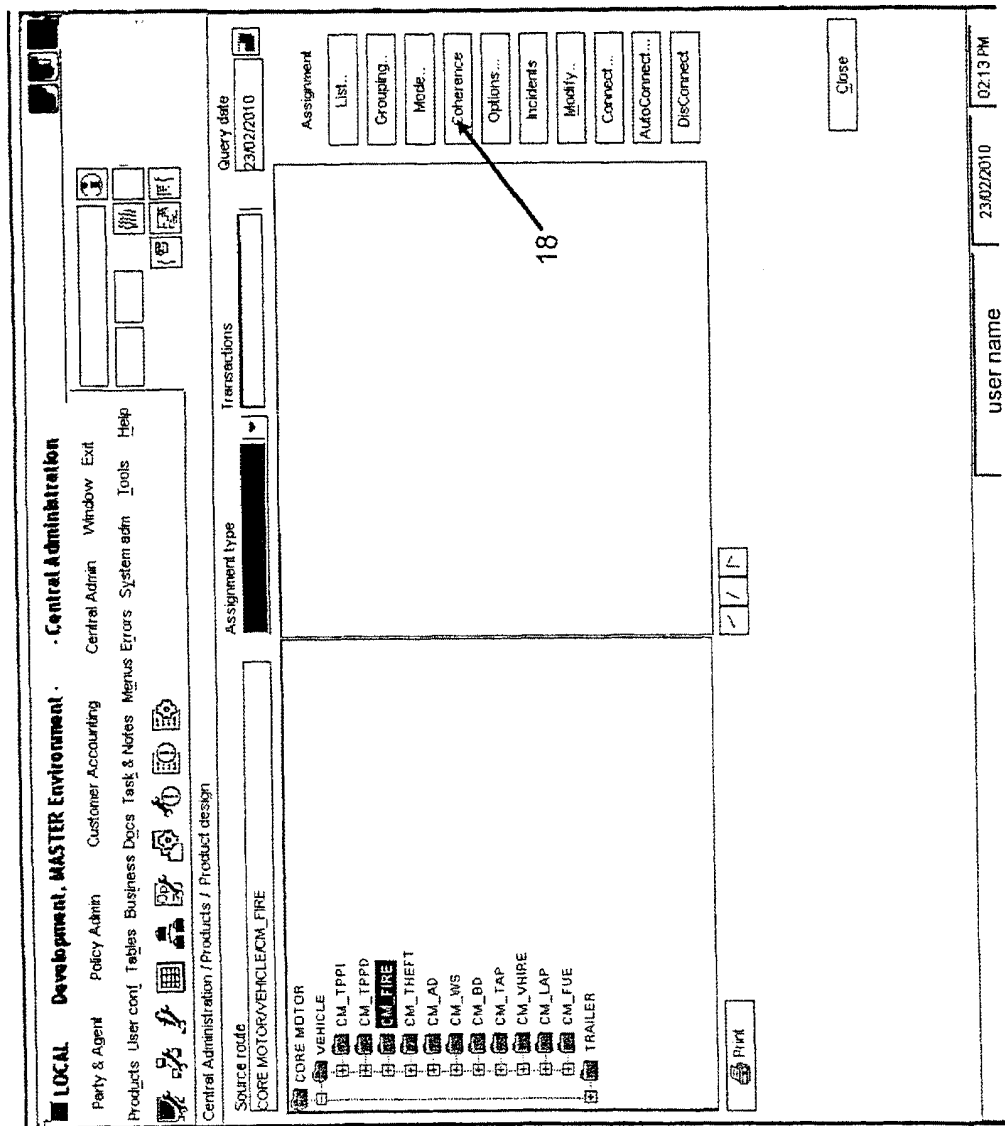


FIG. 27a

Functional Integration 2 - Central Administration

Policy Admin Customer Accounting Central Admin Reinsurance Window Exit
 Products User Conf Tables Business Docs Task & Notes Menus Errors System Admin Tools Help

Central Administration / Products / Coherency Check Window

Template: Street: 
☒ Exclude Coherency Check ☒ Elements coherency ☒ Validations coherency ☒ Ratings coherency

Validation code	Object	Guarantee	Validation type	Item code 1	Item name 1
PRODUCT STRUCTURE COHERENCY			E		
PRODUCT STRUCTURE COHERENCY			E		
ELEMENTS DEFINED ON PF			E		
ELEMENTS DEFINED ON PF			E		
ELEMENTS DEFINED ON DD			E		
ELEMENTS DEFINED ON DD			E		
DRILL DOWN ELEMENTS DEF.			E		
DRILL DOWN ELEMENTS DEF.			E		
PROTECTED ELEMENTS DEF.			E		
PROTECTED ELEMENTS DEF.			E		
RESTRICTIONS TABLES DEF.			E		
RESTRICTIONS TABLES DEF.			E		
STREET CODE DEFINITION			E		
STREET CODE DEFINITION			E		
ELEMENTS LENGTH VALIDATION			E		
ELEMENTS LENGTH VALIDATION			E		

Functional Integration 2 - Central Administration																																																																																							
	Policy Admin Customer Accounting Central Admin Reinsurance Window Exit Products User Cont Tables Business Docs Task & Notes Menus Errors System Adm Tools Help																																																																																						
Template COMBINED SBC_BUS AA -Execute Coherency Check- ☒ Elements coherency ☒ Validations coherency ☒ Ratings coherency Check																																																																																							
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">Validation code</th> <th style="width: 20%;">Object</th> <th style="width: 10%;">Guarantee</th> <th style="width: 10%;">Validation type</th> <th style="width: 10%;">Item code 1</th> <th style="width: 20%;">Item name 1</th> </tr> </thead> <tbody> <tr> <td>OPERATION ELEMENTS DEFINED</td> <td></td> <td></td> <td>R</td> <td>VANETRAT</td> <td>NET RATE EXP</td> </tr> <tr> <td>OPERATION ELEMENTS DEFINED</td> <td></td> <td></td> <td>R</td> <td>VANETRAT</td> <td>NET RATE EXP</td> </tr> <tr> <td>OPERATION ELEMENTS DEFINED</td> <td></td> <td></td> <td>R</td> <td>VANETRAT</td> <td>NET RATE EXP</td> </tr> <tr> <td>OPERATION ELEMENTS DEFINED</td> <td></td> <td></td> <td>R</td> <td>VANETRAT</td> <td>NET RATE EXP</td> </tr> <tr> <td>OPERATION ELEMENTS DEFINED</td> <td></td> <td></td> <td>R</td> <td>VABOOKER</td> <td>VABOOKER</td> </tr> <tr> <td>OPERATION ELEMENTS DEFINED</td> <td></td> <td></td> <td>R</td> <td>VABOOKER</td> <td>VABOOKER</td> </tr> <tr> <td>OPERATION ELEMENTS DEFINED</td> <td></td> <td></td> <td>R</td> <td>VABOOKER</td> <td>VABOOKER</td> </tr> <tr> <td>OPERATION ELEMENTS ON DO</td> <td></td> <td></td> <td>R</td> <td></td> <td></td> </tr> <tr> <td>OPERATIONS NOT DEFINED TWICE</td> <td></td> <td></td> <td>R</td> <td></td> <td></td> </tr> <tr> <td>OPERATIONS NOT DEFINED TWICE</td> <td>Additional Cov...</td> <td>Fidelity</td> <td>R</td> <td>0000031579</td> <td>AD COV COR...</td> </tr> <tr> <td>OPERATIONS NOT DEFINED TWICE</td> <td>Additional Cov...</td> <td>Fidelity</td> <td>R</td> <td>0000031677</td> <td>FIDELITY COR...</td> </tr> <tr> <td>OPERATIONS NOT DEFINED TWICE</td> <td>Trade Pl BR</td> <td>Pl BR - Dinner</td> <td>R</td> <td>0000030006</td> <td>Pl DPL CORR</td> </tr> </tbody> </table>										Validation code	Object	Guarantee	Validation type	Item code 1	Item name 1	OPERATION ELEMENTS DEFINED			R	VANETRAT	NET RATE EXP	OPERATION ELEMENTS DEFINED			R	VANETRAT	NET RATE EXP	OPERATION ELEMENTS DEFINED			R	VANETRAT	NET RATE EXP	OPERATION ELEMENTS DEFINED			R	VANETRAT	NET RATE EXP	OPERATION ELEMENTS DEFINED			R	VABOOKER	VABOOKER	OPERATION ELEMENTS DEFINED			R	VABOOKER	VABOOKER	OPERATION ELEMENTS DEFINED			R	VABOOKER	VABOOKER	OPERATION ELEMENTS ON DO			R			OPERATIONS NOT DEFINED TWICE			R			OPERATIONS NOT DEFINED TWICE	Additional Cov...	Fidelity	R	0000031579	AD COV COR...	OPERATIONS NOT DEFINED TWICE	Additional Cov...	Fidelity	R	0000031677	FIDELITY COR...	OPERATIONS NOT DEFINED TWICE	Trade Pl BR	Pl BR - Dinner	R	0000030006	Pl DPL CORR
Validation code	Object	Guarantee	Validation type	Item code 1	Item name 1																																																																																		
OPERATION ELEMENTS DEFINED			R	VANETRAT	NET RATE EXP																																																																																		
OPERATION ELEMENTS DEFINED			R	VANETRAT	NET RATE EXP																																																																																		
OPERATION ELEMENTS DEFINED			R	VANETRAT	NET RATE EXP																																																																																		
OPERATION ELEMENTS DEFINED			R	VANETRAT	NET RATE EXP																																																																																		
OPERATION ELEMENTS DEFINED			R	VABOOKER	VABOOKER																																																																																		
OPERATION ELEMENTS DEFINED			R	VABOOKER	VABOOKER																																																																																		
OPERATION ELEMENTS DEFINED			R	VABOOKER	VABOOKER																																																																																		
OPERATION ELEMENTS ON DO			R																																																																																				
OPERATIONS NOT DEFINED TWICE			R																																																																																				
OPERATIONS NOT DEFINED TWICE	Additional Cov...	Fidelity	R	0000031579	AD COV COR...																																																																																		
OPERATIONS NOT DEFINED TWICE	Additional Cov...	Fidelity	R	0000031677	FIDELITY COR...																																																																																		
OPERATIONS NOT DEFINED TWICE	Trade Pl BR	Pl BR - Dinner	R	0000030006	Pl DPL CORR																																																																																		

Close

FIG. 27C

Functional Integration 2 - Central Administration

Party & Agent Policy Admin Customer Accounting Central Admin Reinsurance Window Exit

Products User Cont Tables Business Docs Task & Notes Menus Errors System Adm Tools Help

Central Administration / Products / Coherency Check Window

Template Street

☒ Execute Coherency Check

☒ Elements coherency ☒ Validations coherency ☒ Ratings coherency

Validation code	Object	Guarantee	Validation type	Item code 1	Item name 1
OPERATION ELEMENTS DEFINED			R	VADU000001	VADU000001
OPERATION ELEMENTS ON DO			R		
OPERATION ELEMENTS ON DD			R		
OPERATIONS NOT DEFINED TWICE			R		
OPERATIONS NOT DEFINED TWICE	Additional Cov ...	Fidelity	R	0000031579	AD COV COR...
OPERATIONS NOT DEFINED TWICE	Additional Cov ...	Fidelity	R	0000031677	FIDELITY COR...
OPERATIONS NOT DEFINED TWICE	Trade PLPR	PLPR - Proper...	R	0000030986	PL PD LCORE
OPERATIONS NOT DEFINED TWICE	Trade PLPR	PLPR - Proper...	R	0000031603	PRLARGCL O...
OPERATIONS NOT DEFINED TWICE	Clinical Trial Tr ...	Clinical Trial - I...	R	0000031828	NET RATE EXP
OPERATIONS NOT DEFINED TWICE	Clinical Trial Tr ...	Clinical Trial - I...	R	0000032055	TRADE EXPOS
OPERATIONS NOT DEFINED TWICE	Trade EL	EL - Injury	R	0000031050	EL INJ LAR CO...
OPERATIONS NOT DEFINED TWICE	Trade EL	EL - Injury	V	0000031603	PRLARGCL O...
VALIDATIONS CONNECTED			V		
VALIDATIONS CONNECTED			V		
VALIDATION HAS FIRST ELEMENT			V		

user name 19/11/2007 04:00 PM

FIG. 28

