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

(11) PI 0901514-0 B1

(22) Data do Depósito: 27/02/2009

(45) Data de Concessão: 08/10/2019

(54) Título: PRODUTO E SISTEMA, PARA ANÁLISE DE TRANSFORMAÇÃO DE ESCRIPT DE TESTE E MÉTODO PARA TESTAR ANÁLISE DE TRANSFORMAÇÃO DE SCRIPT DE TESTE

(51) Int.Cl.: G06F 11/36.

(30) Prioridade Unionista: 27/02/2008 US 12/038,665.

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

(72) Inventor(es): MARK GRECHANIK; QING XIE; CHEN FU.

(57) Resumo: ANALISADOR DE TRANSFORMAÇÃO DE SCRIPT DE TESTE COM MOTOR DE GUIA DE MUDANÇA. A presente invenção refere-se a um analisador de script com um guia de mudança que gera scripts de teste acurados para a evolução de aplicativos. Os aplicativos frequentemente têm interfaces gráficas de usuário complexas para as quais permutações e combinações de elementos de GUI dão origem a um campo enorme de comandos e sequências de comando em potencial a serem testados. Além disso, estes aplicativos mudam ao longo do tempo, tornando os scripts de teste anteriores não-praticáveis, O analisador de script automaticamente gera novos scripts de teste para se testarem de forma confiável as versões de aplicativo subsequentes, enquanto reduz grandemente o tempo, o custo e os gastos de recurso necessários para se chegar a versões de aplicativo subsequentes.

Relatório Descritivo da Patente de Invenção para **"PRODUTO E SISTEMA, PARA ANÁLISE DE TRANSFORMAÇÃO DE ESCRIPT DE TESTE E MÉTODO PARA TESTAR ANÁLISE DE TRANSFORMAÇÃO DE SCRIPT DE TESTE".**

ANTECEDENTES DA INVENÇÃO

1. Referência Cruzada a Pedidos Relacionados

[001] Este pedido está relacionado aos pedidos a seguir, todos depositados no mesmo dia e todos incorporados inteiramente aqui como referência:

[002] Número de protocolo legal 10022-1162: Número de Série de Pedido de Patente U.S. 12/038.665, depositado em 27 de fevereiro 2008;

[003] Número de protocolo legal 10022-1185: Número de Série de Pedido de Patente U.S. 12/038.672, depositado em 27 de fevereiro 2008;

[004] Número de protocolo legal 10022-1186: Número de Série de Pedido de Patente U.S. 12/038.676, depositado em 27 de fevereiro 2008;

[005] Número de protocolo legal 10022-1187: Número de Série de Pedido de Patente U.S. 12/038.661, depositado em 27 de fevereiro 2008;

[006] Número de protocolo legal 10022-1188: Número de Série de Pedido de Patente U.S. 12/038.658, depositado em 27 de fevereiro 2008; e

[007] Número de protocolo legal 10022-1189: Número de Série de Pedido de Patente U.S. 12/038.675, depositado em 27 de fevereiro 2008.

2. Campo Técnico

[008] Esta exposição refere-se à análise e à geração de scripts de teste para se testarem aplicativos de interface gráfica de usuário e,

em particular, refere-se à transformação de um script de teste anterior para uso com uma nova versão de aplicativo.

3. Técnica Relacionada

[009] O passo assíduo da tecnologia avançando de origem a aplicativos de software de computador complexos para ajudarem na automação de quase todo aspecto da existência do dia a dia. Os aplicativos de hoje em dia existem para ajudarem com a redação de romances, para preenchimento de restituição de imposto de renda, para análise de tendências históricas em nomes de bebês. Um recurso quase ubíquo destes aplicativos é que eles empregam interfaces gráficas de usuário (GUIs). Um outro aspecto quase ubíquo é que os aplicativos de GUI (GAPs) requerem testes completos antes da liberação.

[0010] Não obstante, no passo, era mais fácil implementar a GUI para o aplicativo do que testar completamente o GAP. Para GAPs de qualquer complexidade significativa, as permutações e combinações de elementos de GUI dão margem a um campo enorme de comandos potenciais e sequências de comando que poderiam ter bugs de qualquer severidade, de uma falha insignificante a uma crítica. A exacerbação do problema é que os desenvolvedores de aplicativo estão sob pressão para continuamente adicionarem novos recursos, atualizarem a GUI e liberarem novas versões de aplicativo. Como resultado, mesmo se um script passado de uma versão anterior de um GAP fosse adequado, raramente é o caso de o script de teste original poder testar adequadamente o aplicativo revisado subsequente.

[0011] Testar manualmente GAPs de empresa em larga escala é tedioso, propenso a erros e trabalhoso. Os GAPs não triviais contêm centenas de telas de GUI que, por sua vez, contêm milhares de objetos de GUI. De modo a se automatizarem os testes de GAPs, os engenheiros de teste escrevem programas usando linguagens de

script (por exemplo, JavaScript e VBScript), e estes scripts de teste comandam os GAPs através de estados diferentes ao imitarem usuários que interagem com estes GAPs pela realização de ações nos seus objetos de GUI. Frequentemente, os scripts de teste simulam usuários de GAPs e suas declarações acessam e manipulam objetos de GUI destes GAPs. Por exemplo, a declaração:

[0012] VbWindow("Login").VbEdit("txtAgentsName").Set "Shawn"

[0013] localiza uma janela cujo título descritivo é Login e que é criada por um controle baseado em Visual Basic, então, localiza uma caixa de texto cujo nome é txtAgentsName que é um objeto de GUI cuja origem é a janela de login. Ao chamar o método Set com o parâmetro "Shawn", o valor da caixa de texto é regulado para "Shawn".

[0014] Ferramentas comerciais tais como QTP e Rational Robot ajudam na geração de scripts de teste pelo acompanhamento do apontamento de um cursor em objetos de GUI e realizando ações desejadas. Estas ferramentas geram um código de script que pode reexecutar ações de usuário capturadas. O código gerado serve como um esqueleto para a criação de scripts para a automação de testes de script. Os engenheiros de teste adicionam um código aos scripts gerados de modo que estes scripts possam ser reexecutados usando-se valores de entrada diferentes, desse modo se exercitando a funcionalidade do GAP.

[0015] A expansão de scripts de teste com um código escrito manualmente para automação de testes torna o script de teste mais complexo, difícil de entender, manter e evoluir. Embora seja conhecido de antemão que os scripts de teste acessem e manipulem elementos de GUI, não é claro como detectar as operações no momento da compilação que levem a erros de tempo de execução. O uso de chamadas de API exportadas por plataformas de teste permanece um modo primário de acesso e manipulação de objetos de GUI de GAPs,

e estas chamadas de API levam a vários erros de tempo de execução nos scripts de teste. Por exemplo, o pessoal de teste pode usar as chamadas de API de plataforma forma incorreta no código fonte de script de teste, desse modo acessando elementos de GUI que eles não pretendiam acessar.

[0016] É um desafio técnico difícil checar scripts de teste quanto a potenciais falhas causadas por chamadas de API de terceiros que levam a testes incorretos e erros de tempo de execução nos scripts de teste. Além disso, há problemas fundamentais com o uso de chamadas de API para acesso e manipulação de objetos de GUI. Em primeiro lugar, as chamadas de API assumem nomes e valores de propriedade de objetos de GUI como variáveis de parâmetro de entrada de string. Os valores destes parâmetros de entrada frequentemente são conhecidos apenas no tempo de execução, tornando impossível aplicar algoritmos de checagem bons. Em segundo lugar, as plataformas de teste exportam dúzias de chamadas de API diferentes, e a alta complexidade destas chamadas de API torna difícil para os programadores entenderem as chamadas de API a usar e como combiná-las para acesso e manipulação de objetos de GUI. Estes problemas levam a uma ampla faixa de bugs nos scripts de teste, muitos dos quais sendo difíceis de detectar durante a inspeção do código fonte de script de teste.

[0017] Um problema adicional surge porque as especificações de exigência de aplicativo incluem conceitos de nível alto que descrevem GAPs, especificamente, seus objetos de GUI. Infelizmente, o rastreio de objetos de GUI de GAPs para estes conceitos de nível alto é um problema difícil porque os programadores não documentam estes rastreios. Assim sendo, quando o pessoal de teste cria GAPs, eles gastam um tempo considerável para entenderem como usar estes GAPs pela leitura de documentação e falando com especialistas no

assunto. Este conhecimento crucial frequentemente é perdido após o pessoal de teste ser reatribuído a outras tarefas ou deixarem a companhia.

[0018] Um dos benefícios percebidos de abordagens existentes para a criação de scripts de teste é que uma checagem de tipo não é requerida, uma vez que o código de script é gerado diretamente a partir de GUIs. Por exemplo, dados certos objetos de GUI em um GAP, uma ferramenta de teste pode produzir declarações correspondentes que navegam para aqueles objetos usando chamadas de API com parâmetros de string descrevendo suas propriedades. Contudo, este benefício percebido de fato dá margem a desafios técnicos difíceis, devido a inconsistências semânticas entre o script de teste e o GAP. Suponha, por exemplo, que durante a fase de manutenção a GUI do GAP mudasse. O intérprete de script não está ciente da mudança e rodaria o script gerado sem a produção de quaisquer avisos de tempo de compilação. Contudo, o script resultante falharia no tempo de execução ou produziria resultados de teste inconsistentes porque seu código tentaria acessar objetos de GUI que mudaram ou não existem mais.

[0019] Portanto, existe uma necessidade de uma arquitetura de geração de script de teste com análise de suporte e lógica de avaliação que se dirija aos problemas citados acima e a outros encontrados previamente.

SUMÁRIO

[0020] Uma arquitetura de transformação de script de teste gera scripts de teste acurados para aplicativos com interfaces gráficas de usuário que mudam ao longo do tempo. Conforme os aplicativos mudam, scripts de teste prévios são tornados não praticáveis. A arquitetura facilita a geração automática de novos scripts de teste para se testarem de forma confiável versões de aplicativo subsequentes e

pode reduzir grandemente o tempo, o custo e as despesas de recurso necessárias para se chegar a novos scripts de teste.

[0021] A arquitetura de transformação de script de teste ("arquitetura") pode incluir um modelo de tipo de interface gráfica de usuário (GUI) e uma lógica de construtor de modelo de GUI. A lógica de construtor de modelo de GUI aceita como entrada o modelo de tipo de GUI, uma versão de GAP atual e uma versão de GAP subsequente. A lógica de construtor de modelo de GUI gera como uma saída um modelo de GUI de GAP atual e um modelo de GUI de GAP subsequente.

[0022] Além disso, a arquitetura inclui uma lógica de comparador de GUI. A lógica de comparador de GUI aceita como entrada o modelo de GUI de GAP atual, o modelo de GUI de GAP subsequente e gera como saída um modelo de diferença de GUI. O modelo de diferença de GUI inclui entradas de diferença de elemento de GUI que identificam um elemento de GUI específico que combina entre a versão de GAP atual e a versão de GAP subsequente, mas que difere no caráter entre a versão de GAP atual e a versão de GAP subsequente. A arquitetura ainda inclui um analisador de script. Com base em uma representação de árvore de sintaxe abstrata de um script de teste atual, o analisador de script gera um guia de mudança, um script de teste transformado ou ambos. O guia de mudança pode incluir, por exemplo, uma informação de transformação de script para transformação do script de teste atual para uso em relação à versão de GAP subsequente. O script de teste transformado (para uso em relação à versão de GAP subsequente) pode incluir, por exemplo, entradas de script modificadas geradas a partir das entradas de script de teste atuais no script de teste atual.

[0023] Outros sistemas, métodos, recursos e vantagens serão ou se tornarão evidentes para alguém versado na técnica, mediante um

exame das figuras a seguir e da descrição detalhada. Todos esses sistemas, métodos, recursos e vantagens adicionais estão incluídos nesta descrição, estão no escopo do assunto reivindicado e são protegidos pelas reivindicações a seguir.

BREVE DESCRIÇÃO DOS DESENHOS

[0024] O sistema pode ser mais bem entendido com referência aos desenhos e à descrição a seguir. Os elementos nas figuras não estão necessariamente em escala, a ênfase sendo posta, ao invés disso, na ilustração dos princípios do sistema. Nos desenhos, números referenciados iguais designam partes correspondentes por todas as diferentes vistas.

[0025] A Figura 1 mostra uma arquitetura de transformação de script de teste.

[0026] A Figura 2 mostra um fluxograma de processamento realizado pela arquitetura de transformação de script de teste.

[0027] A Figura 3 mostra um sistema de tipificação de elemento de GUI.

[0028] A Figura 4 mostra um mapeamento de tipo de elemento de GUI.

[0029] A Figura 5 mostra uma interface de usuário de mapeamento de tipo de elemento de GUI.

[0030] A Figura 6 mostra um fluxograma para uma lógica de tipificação de elemento de GUI.

[0031] A Figura 7 mostra um sistema de mapeamento de elemento de GUI.

[0032] A Figura 8 mostra um mapeamento de versão de elemento de GUI.

[0033] A Figura 9 mostra uma interface de usuário de mapeamento de versão de elemento de GUI.

[0034] A Figura 10 mostra um fluxograma para uma lógica de

mapeamento de versão de elemento de GUI.

[0035] A Figura 11 mostra um sistema de tipificação e de mapeamento de elemento de GUI.

[0036] A Figura 12 mostra uma arquitetura de ferramenta de evolução de metadados.

[0037] A Figura 13 mostra uma arquitetura de depósito de metadados.

[0038] A Figura 14 mostra os exemplos de mensagens de notação.

[0039] A Figura 15 mostra um fluxograma de processamento de metadados.

[0040] A Figura 16 mostra um fluxograma de processamento de tipo.

[0041] A Figura 17 mostra uma primeira parte de um fluxograma de processamento de mapeamento.

[0042] A Figura 18 mostra uma segunda parte de um fluxograma de processamento de mapeamento.

[0043] A Figura 19 mostra um fluxograma de processamento de notação.

[0044] A Figura 20 mostra um fluxograma de processamento de migração de metadados.

[0045] A Figura 21 mostra um fluxograma de processamento de migração de metadados.

[0046] A Figura 22 mostra um fluxograma de processamento de migração de metadados.

[0047] A Figura 23 mostra um sistema de comparador de GAP.

[0048] A Figura 24 mostra uma porção de um modelo de GUI de GAP atual que provê uma representação de GAP como um modelo para análise.

[0049] A Figura 25 mostra uma porção de um modelo de GUI de

GAP subsequente que provê uma representação de GAP como um modelo para análise.

[0050] A Figura 26 mostra uma porção de um modelo de diferença de GUI.

[0051] A Figura 27 mostra um fluxograma para uma lógica de construtor de modelo de GUI.

[0052] A Figura 28 mostra um fluxograma para uma lógica de comparação de GAP.

[0053] A Figura 29 mostra um visor e uma interface de limite de comparador de GUI.

[0054] A Figura 30 mostra um destaque de elemento de GUI em resposta a um cursor deslizante de limite de comparador.

[0055] A Figura 31 mostra um fluxograma para uma lógica de visualização.

[0056] A Figura 32 mostra um exemplo de propriedades de bissimulação.

[0057] A Figura 33 mostra uma GUI de uma versão de GAP atual.

[0058] A Figura 34 mostra uma GUI de uma versão de GAP subsequente.

[0059] A Figura 35 ilustra uma comparação de GAP atual e subsequente.

[0060] A Figura 36 ilustra uma entrada de diferença de elemento de GUI.

[0061] A Figura 37 mostra um script de teste atual.

[0062] A Figura 38 mostra uma representação de script de teste atual.

[0063] A Figura 39 mostra um sistema analisador de script de exemplo.

[0064] A Figura 40 mostra um fluxograma para a recuperação de propriedades de uma entrada de objeto de GUI a partir de um depósito

de objeto (OR).

[0065] A Figura 41 mostra um fluxograma para a identificação de uma entrada de diferença de GUI correspondente a uma entrada de objeto de GUI.

[0066] A Figura 42 mostra um script de teste transformado.

[0067] A Figura 43 mostra um guia de mudança.

[0068] A Figura 44 mostra um fluxograma para extração de declarações de script de teste transformado.

[0069] A Figura 45 ilustra uma outra entrada de diferença de elemento de GUI.

[0070] A Figura 46 mostra uma outra entrada de diferença de elemento de GUI de exemplo.

[0071] A Figura 47 ilustra caminhos de navegação a partir de uma fonte para um objeto de destino.

[0072] A Figura 48 mostra um fluxograma para a extração de um especificador de mudança de GAP.

[0073] A Figura 49 mostra um analisador de transformação de script de teste com arquitetura de motor de custo econômico.

[0074] A Figura 50 mostra um script de teste atual.

[0075] A Figura 51 mostra uma representação de script de teste atual.

[0076] A Figura 52 mostra um sistema de motor de custo econômico de exemplo.

[0077] A Figura 53 mostra um fluxograma para a identificação de uma entrada de diferença de GUI correspondente a uma entrada de objeto de GUI.

[0078] A Figura 54 mostra um script de teste transformado.

[0079] A Figura 55 mostra um fluxograma para a geração de um especificador de mudança de GAP sintético.

[0080] A Figura 56 mostra um fluxograma para a extração de um

relatório de custo de transformação de script de teste.

DESCRIÇÃO DETALHADA DAS MODALIDADES PREFERIDAS

[0081] A Figura 1 mostra uma arquitetura de transformação de script de teste 100 ("arquitetura 100"). A arquitetura 100 inclui vários sistemas, incluindo um sistema de tipificação e de mapeamento de elemento de GUI 102, um sistema de migração e de depósito de metadados 104 e um sistema de comparador de aplicativo de GUI (GAP) 106. A arquitetura 100 também inclui um sistema de análise de script de teste 108 e um sistema de análise econômica de script de teste 110.

[0082] O sistema de tipificação e de mapeamento de elemento de GUI 102 ("sistema 102") facilita a aplicação de um modelo de tipo formal a elementos de GUI em GAPs, bem como o estabelecimento de relações entre elementos de GUI de uma versão de um GAP para uma outra. O sistema 102 inclui um processador 112, uma memória 114 e um depósito de dados de elemento de GUI ("depósito") 116. O sistema 102 troca informação com os outros sistemas 104, 106, 108 e 110 na arquitetura 100 através da lógica de comunicação 118.

[0083] A memória 114 mantém a lógica de tipificação e de mapeamento de GUI 120 e um modelo de tipo de GUI de referência 122. O depósito 116 armazena mapeamentos de elemento de GUI de aplicativo de interface gráfica de usuário (GAP) ("mapeamentos") 124. O depósito 116 também armazena um modelo de tipo de GUI de versão de GAP atual 126 e um modelo de tipo de GUI de versão de GAP subsequente 128. O sistema 102 comunica mensagens de especificação de tipo de elemento de GUI e mensagens de especificação de mapeamento de elemento de GUI para o sistema de migração e de depósito de metadados 104 ("sistema 104").

[0084] O sistema 104 facilita a perpetuação de metadados úteis para GAPs, particularmente conforme os GAPs evoluírem entre

versões. O sistema 104 inclui um processador 129, uma memória 130 que mantém uma lógica de processamento de metadados 132 e um sistema de gerenciamento de banco de dados 134 e uma lógica de comunicação 136. O sistema 104 também inclui um depósito de metadados 138. O depósito de metadados 138 mantém metadados de elemento de GUI 140, tais como identificadores de elemento de GUI 142, notações de elemento de GUI 144, tipos de elemento de GUI 146 e mapeamentos de elemento de GUI 148. O sistema 104 pode comunicar os metadados de elemento de GUI 140 para os outros sistemas 102, 106, 108 e 110 para ajudar aos outros sistemas 102, 106, 108 e 110 na execução de sua lógica.

[0085] O sistema comparador de GAP 106 ("o sistema 106") analisa uma versão de GAP atual 150 e uma versão de GAP subsequente 152 para determinar diferenças de elemento de GUI entre as versões 150 e 152. Para essa finalidade, o sistema 106 pode incluir uma lógica de construtor de modelo de GUI 154 que aceita como entrada a versão de GAP atual 150 e a versão de GAP subsequente 152. A lógica de construtor de modelo de GUI 154 gera um modelo de elemento de GUI 156 da versão de GAP atual 150 e um modelo de elemento de GUI 158 da versão de GAP subsequente 152. Os modelos 156 e 158 podem ser modelos de árvore ou outros tipos de representações.

[0086] Um comparador de GUI 160 processa as representações 156 e 158. Um modelo de diferença de GUI 162 resulta. O modelo de diferença de GUI 162 inclui entradas de diferença de elemento de GUI que identificam um elemento de GUI específico que combina entre a versão de GAP atual e a versão de GAP subsequente, mas que difere no caráter entre a versão de GAP atual e a versão de GAP subsequente. Por exemplo, uma entrada de diferença pode identificar que uma caixa de texto na versão de GAP atual combina com uma

lista suspensa na versão de GAP subsequente, e pode notar as características de elemento de GUI diferentes entre as duas versões.

[0087] O sistema de análise de script de teste 108 ("sistema 108") provê um analisador de transformação de script de teste com um motor de guia de mudança. Em particular, o sistema 108 começa com um script de teste atual 164 e inicia uma execução de um analisador gramatical de script 166 para a obtenção de uma representação de árvore de sintaxe abstrata 168 ("AST 168") do script de teste atual 164. O script de teste atual 164 pode ser adequado para se testar qualquer quantidade de funcionalidade na versão de GAP atual 150. O sistema 108 ajuda a transformar o script de teste atual 164 em um script de teste transformado adequado para se testar qualquer quantidade de funcionalidade na versão de GAP subsequente 152.

[0088] O analisador de script 170 aceita a AST 168, o modelo de diferença de GUI 162 e os metadados de elemento de GUI 140 através da lógica de comunicação 190. A lógica de consulta de depósito de objeto 172 recupera dados de objeto a partir do depósito de objeto 174 como um auxílio na análise do modelo de diferença de GUI 162. Como resultado desta análise, o analisador de script 170 gera um script de teste transformado 178, um guia de mudança 180 ou ambos. A análise pode levar em consideração um depósito de mensagem de guia de mudança 192 e regras de mudança de script de elemento de GUI 194. Um aspecto adicional do sistema 108 é o motor de satisfação de restrição 188. Em conjunto com a informação de tipo de elemento de GUI obtida a partir dos sistemas 104 e 104, o motor de satisfação de restrição determina se as declarações de script são compatíveis com o tipo de elemento de GUI atribuído a qualquer dado elemento de GUI.

[0089] O sistema de análise econômica de script de teste 110 ("sistema 110") provê um analisador de transformação de script de

teste com um motor de custo econômico. O sistema 10 inclui um motor econômico 182 que consulta modelos econômicos 176 para a geração de relatórios de custos 186. Os modelos econômicos 176 podem ser modelos econômicos predefinidos de relações de custo de transformação de script de teste. Os relatórios de custos 186 podem incluir uma informação de custo de transformação de script de teste, por exemplo, o custo estimado para transformação do script de teste atual 164 no script de teste transformado 178. Os relatórios de custos 186 podem ser baseados em uma análise do guia de mudança 180, especificamente de especificadores de mudança de GAP de entrada 184, ou uma outra informação.

[0090] A Figura 2 mostra um fluxograma 200 de processamento realizado pela arquitetura de transformação de script de teste 100. O sistema 102 gera mensagens de especificação de mapeamento de elemento de GUI que podem incluir um mapeamento de versão de elemento de GUI, e mensagens de especificação de tipo que podem incluir identificadores de tipo de elemento de GUI (202). O sistema 102 comunica as mensagens para o sistema 104 (204), o qual recebe as mensagens. O sistema 104 em resposta mantém metadados de elemento de GUI no depósito de metadados 138 (206).

[0091] O sistema 106 inicia uma execução de lógica de comparador de GUI (208). A lógica de comparador de GUI 160 aceita como entrada um modelo de elemento de GUI 156 para a versão de GAP atual 150 e um modelo de elemento de GUI 158 para a versão de GAP subsequente 152 (210). A lógica de comparador de GUI 160 ainda pode aceitar como entrada mapeamentos de elemento de GUI a partir do sistema 102 ou 104. A lógica de comparador de GUI 160 produz como saída um modelo de diferença de GUI 162 que inclui entradas de diferença de elemento de GUI que identificam elementos de GUI específicos que combinam entre a versão de GAP atual 150 e

a versão de GAP subsequente 152, mas diferem no caráter entre a versão de GAP atual 150 e a versão de GAP subsequente 152 (212).

[0092] O sistema de análise de script de teste 108 inicia a execução de um analisador de script 170 (214). O analisador de script 170 aceita como entrada o modelo de diferença de GUI 162 e a AST 168 do script de teste atual 164. O analisador de script 170 pode gerar um guia de mudança 180 que inclui uma informação de transformação de script para transformação do script de teste atual 164 para uso em relação à versão de GAP subsequente 152 (216). De forma adicional ou alternativa, o analisador de script 170 pode gerar um script de teste transformado 178 para uso em relação à versão de GAP subsequente 152 (218). O script de teste transformado 178 inclui entradas de script de teste transformado geradas a partir de entradas de script de teste atual no script de teste atual.

[0093] Além disso, o sistema 110 pode iniciar uma execução de um motor econômico 182 operável para analisar o guia de mudança 180 (220). O sistema 100 leva em consideração as relações de custo de transformação de script de teste definidas nos modelos econômicos 176. O resultado é um relatório de custo 186 compreendendo uma informação de custo de transformação de script de teste (222).

[0094] A Figura 1 mostra um sistema de tipificação e de mapeamento de elemento de interface gráfica de usuário (GUI) ("sistema") 102. O sistema 102 inclui um processador 112, uma memória 114 e um depósito de dados de elemento de GUI ("depósito") 116. O sistema 102 troca informação com os outros sistemas através da lógica de comunicação 118. A lógica de comunicação 118 pode ser uma interface com fio/ sem fio, um mecanismo de comunicação interprocesso, uma memória compartilhada, uma interface de serviços da web, ou quaisquer outros tipos de interface de comunicação.

[0095] A memória 114 mantém a lógica de tipificação e de

mapeamento de GUI 120 e um modelo de tipo de GUI de referência 122. Conforme explicado em maiores detalhes abaixo, a lógica de tipificação e de mapeamento de GUI 120 ajuda o projetista de GAP a especificar tipos de elemento de GUI para elementos de GUI individuais de um GAP. A lógica de tipificação e de mapeamento de GUI 120 também ajuda o projetista de GAP a definir links para elementos de GUI em uma versão de GAP atual para elementos de GUI em uma versão de GAP subsequente. Os tipos e a informação de mapeamento podem ser providos para outros sistemas de análise de GAP, tais como sistemas de análise de script de teste e depósitos de metadados, através da lógica de comunicação 118.

[0096] O sistema 102 pode operar em qualquer elemento de GUI em particular. Os exemplos de elementos de GUI incluem caixas de texto, menus, itens de menu, botões de rádio, caixas de verificação, e botões de comando. Outros exemplos incluem caixas de lista, caixas de combinação, botões de alternar, botões de girar, barras de rolagem, rótulos, barras de ferramenta, widgets, imagens, janelas, calendário e *tab strips*. O modelo de tipo de GUI de referência 122 pode estabelecer um conjunto formalizado de nomes de tipo (por exemplo, identificadores) e qualidades comuns a elementos de GUI individuais que distinguem os elementos de GUI do mesmo tipo como membros de uma classe identificável. O modelo de tipo de GUI 122, em conjunto com a especificação de tipos para elementos de GUI individuais ajuda a prover uma técnica sintática tratável para o estabelecimento que a GUI não exibe certos comportamentos (por exemplo, armazenamento de um caractere alfabético em uma caixa de texto significando manter um número de seguridade social), em oposição a uma anotação *ad hoc* de forma livre de elementos de GUI.

[0097] O depósito 116 armazena mapeamentos de elemento de GUI de aplicativo de interface gráfica de usuário (GAP)

("mapeamentos") 124. O depósito 116 também armazena um modelo de tipo de GUI de versão de GAP atual 126 e um modelo de tipo de GUI de versão de GAP subsequente 128. O sistema 102 pode preparar os mapeamentos 124 em resposta a uma entrada de operador que cria links de elementos de GUI a partir de qualquer versão de GAP atual para qualquer versão de GAP subsequente. Assim, por exemplo, se uma caixa de texto mudar para uma lista suspensa, o operador poderá documentar a mudança ao criar explicitamente um link dos dois elementos de GUI em conjunto. O sistema 102 em resposta prepara um mapeamento de versão de elemento e armazena o mapeamento de versão de elemento no depósito 116.

[0098] De modo similar, o sistema 102 pode construir os modelos de tipo 126 e 128 a partir de atribuições específicas de operador de tipos de meio de gancho para elementos de GUI específicos na versão de GAP atual e na versão de GAP subsequente. Para essa finalidade, o sistema 102 pode alertar o operador para uma seleção a partir de uma lista de tipo de elemento de GUI estabelecida pelo modelo de tipo de GUI de referência 122. Cada modelo de tipo 126 e 128 pode incluir especificadores de tipo de elemento de GUI para um ou mais elementos no GAP atual e no GAP subsequente. O sistema 102 pode interagir com o operador usando o visor 121 e também pode aceitar uma linha de comando, um script, um arquivo de batch e outros tipos de entrada.

[0099] A Figura 3 mostra uma implementação do sistema 102. A Figura 3 mostra que a lógica de comunicação 118 pode trocar uma informação com o depósito de metadados 202 através de uma conexão de uma ou mais redes 304. Como exemplos, o sistema 102 pode comunicar mensagens de especificação de tipo de elemento de GUI e mensagens de especificação de mapeamento de elemento de

GUI para o depósito de metadados 202.

[00100] Um detalhe adicional do modelo de tipo de GUI de referência 122 é mostrado na Figura 3. O modelo de tipo de GUI de referência 122 inclui entradas de modelo de tipo de GUI (por exemplo, as entradas de modelo de tipo de GUI 306 e 308). As entradas de modelo de tipo de GUI definem um sistema de tipo para uma GUI, e podem prover um conjunto finito de identificadores preestabelecidos para etiquetagem de elementos de GUI. Cada entrada de modelo de tipo de GUI pode especificar um especificador de tipo de elemento de GUI 310, restrições de modificação de elemento de GUI 312 e restrições de conteúdo de elemento de GUI 314. Em outras implementações, o modelo de tipo de GUI de referência 122 pode incluir uma informação adicional, diferente ou menor. O especificador de tipo de elemento de GUI estabelece um identificador (por exemplo, uma string única ou um número) que pode ser atribuído a um elemento de GUI para especificação de um tipo de elemento de GUI para o elemento de GUI. As restrições de modificação de elemento de GUI 312 especificam como e se um elemento de GUI do tipo especificado pode ser modificado. De modo similar, as restrições de conteúdo e de formatação de elemento de GUI 314 especificam que conteúdo um elemento de GUI do tipo especificado pode manter, e como o conteúdo ou o elemento de GUI em si pode ser formatado. As restrições 312 e 314 podem ser expressas de muitas formas, tal como por regras que ditam o comportamento desejado ou restrições em um elemento de GUI de um dado tipo.

[00101] Os exemplos de tipos de elemento de GUI são dados abaixo na Tabela 1.

Tabela 1		
Especificador de Tipo de Elemento de GUI	Restrições de Modificação de Elemento de GUI	Restrições de Conteúdo e Formatação de Elemento de GUI
SSNEntryBox	nenhuma	Nove dígitos, ou 11 caracteres da forma: ###-##-#### onde # é um dígito e '-' é o símbolo de traço.
SSNDisplayBox	Apenas Leitura	Nove dígitos, ou 11 caracteres da forma: ###-##-#### onde # é um dígito e '-' é o símbolo de traço.
StaticWindowLabel	Apenas Leitura	0 a 50 caracteres alfanuméricos
3DButton	nenhuma	Cor de Primeiro Plano Faixa: FStart - FEnd Cor de Fundo Faixa: BStart - BEnd
HelpTextEntryBox	nenhuma	faixa de posição X: XMin: XMax faixa de posição Y: YMin: YMax FontSize: 10 pontos no mínimo 16 pontos no máximo
MenuWidget	localização X, Y fixa	Tamanho Mínimo = SMin Tamanho Máximo = SMax
US-StateTextBox	nenhuma	Apenas os nomes dos estados dos Estados Unidos são permitidos.

[00102] Na Tabela 1, o especificador SSNEntryBox pode ser anexado, por exemplo, a um elemento de GUI (por exemplo, uma caixa de texto) em um GAP que é pretendido para entrada de números de seguridade social. O tipo SSNEntryBox não impõe restrições nas modificações que podem ser realizadas na caixa de texto. Contudo, o tipo SSNEntryBox restringe o conteúdo da caixa de texto para nove dígitos ou 3 dígitos seguidos por um traço, seguido por 2 dígitos,

seguidos por um traço, seguido por 4 dígitos. O tipo SSNDisplayBox especifica restrições similares, mas também torna o elemento de GUI apenas de leitura.

[00103] O tipo StaticWindowLabel provê um rótulo de apenas leitura para seu elemento de GUI anexado. O rótulo pode ter entre 0 e 50 caracteres alfanuméricos. Devido ao fato de StaticWindowLabel ser do tipo apenas de leitura de elemento de GUI, o rótulo não pode ser mudado.

[00104] O tipo 3DButton limita as cores de primeiro plano que podem ser reguladas para o elemento de GUI entre FStart e FEnd. De modo similar, o tipo 3DButton limita as cores de fundo que podem ser reguladas para o elemento de GUI entre BStart e BEnd. A HelpTextEntryBox restringe a posição X e Y do elemento de GUI anexado, e ainda restringe a fonte dimensionada usada para o texto no elemento de GUI entre 10 e 16 pontos. O tipo MenuWidget pode ser aplicado a um elemento de botão gráfico, por exemplo, para o estabelecimento do elemento como um widget para uma barra de menu. O tipo MenuWidget fixa a localização X e Y do elemento de GUI, e ainda estabelece um tamanho mínimo de SMin e um tamanho máximo de SMax. A US-StateTextBox limita o conteúdo de um elemento de GUI a apenas os nomes dos Estados dos Estados Unidos (por exemplo, "Alaska, Maine, Nevada, ...").

[00105] O sistema 102 pode implementar qualquer modelo de tipo de GUI formal 122 útil para ajudar um projetista a criar um GAP. O modelo de tipo de GUI pode variar, dependendo do tipo de GAP sendo projetado, e o sistema pode escolher a partir de múltiplos modelos de tipo de GUI 122 para o GAP sob análise, ou pode aceitar uma entrada de operador para selecionar qual modelo de tipo de GUI é aplicável. Por exemplo, o operador pode especificar um modelo de tipo de GUI de assistência médica, quando da construção de um GAP de farmácia,

e especificar um modelo de tipo de GUI de videogame, quando construindo uma interface de usuário para um jogo on-line de RPG. O modelo de tipo de GUI de assistência médica pode incluir tipos de elemento úteis para a construção de um GAP relacionado à assistência médica, tal como o tipo SSNEntryBox, enquanto o modelo de tipo de GUI de videogame pode incluir tipos de elemento úteis para a construção da interface de usuário, tal como o 3DButton.

[00106] A Figura 3 também mostra que o sistema 102 inclui a lógica de tipificação de elemento de GUI ("lógica de mapeamento de tipo") 316 residente na memória 114. A lógica de mapeamento de tipo 316 pode ser incluída na lógica de tipificação e de mapeamento de GUI 120, mas também pode ser um módulo separado ou implementada de outras formas. Conforme descrito em detalhes abaixo, a lógica de mapeamento de tipo 316 pode incluir uma lógica de seleção de elemento 318, uma lógica de recuperação de tipo 320 e uma lógica de seleção de tipo 326. A lógica de tipificação 316 ainda pode incluir uma lógica de associação de tipo 324 e uma lógica de criação de mensagem 326. A lógica de mapeamento de tipo 316 gera mapeamentos de tipo de elemento de GUI (por exemplo, o mapeamento de tipo 328).

[00107] Voltando-nos brevemente para a Figura 4, um exemplo de um mapeamento de tipo de elemento de GUI 400 é mostrado. O mapeamento de tipo 400 inclui um alias de GAP 402, um identificador de elemento de GUI 404, e um identificador de tipo de GUI 406. Adicionalmente, menos campos ou diferentes podem ser incluídos no mapeamento de tipo 400.

[00108] O alias de GAP 402 especifica um identificador para o GAP, o qual inclui o elemento de GUI ao qual um tipo está sendo aplicado. O alias de GAP 402 pode ser um identificador único que distingue entre GAPs, incluindo uma versão de GAP atual e uma versão subsequente

do mesmo GAP. O identificador de elemento de GUI 404 provê um identificador único para o elemento de GUI o qual está sendo tipificado. O identificador de tipo de GUI 406 especifica o tipo de elemento de GUI sendo atribuído ao elemento de GUI (por exemplo, SSNEntryBox).

[00109] A Figura 4 também mostra dois exemplos de mapeamentos de tipo de GUI 408 e 410. O mapeamento de tipo 408 é um mapeamento para um elemento de GUI de Window. O alias de GAP 412 é "University Directory0", significando a versão atual de um GAP de diretório de universidade. O elemento de GUI sendo tipificado tem o identificador de elemento único 414 "0x30fb0" notado pelo identificador de HWND e estabelecido, por exemplo, por uma interface de camada de acessibilidade, conforme descrito abaixo.

[00110] O identificador de tipo de GUI 416 para o elemento de GUI Window é "US-StateTextBox". O mapeamento de tipo 410 é um mapeamento para um elemento de GUI de Menu Item. O alias de GAP 418 é "University Directory1", significando a versão subsequente do GAP de diretório de universidade. O elemento de GUI sendo tipificado tem o identificador de elemento único 420 "OpenFile" conforme especificado no campo Name. O identificador de tipo de GUI 422 para o elemento de GUI Window é "FileOpen".

[00111] A Figura 4 também mostra um exemplo de uma mensagem de especificação de tipo de elemento de GUI 424. A mensagem de especificação de tipo de elemento de GUI 424 pode incluir um cabeçalho de mensagem de especificação de tipo de elemento de GUI 426 e um terminador de mensagem de especificação de tipo de elemento de GUI 428. O cabeçalho 426 e o terminador 428 significam que os dados na mensagem especificam um mapeamento de tipo para um elemento de GUI. Para essa finalidade, a mensagem de especificação de tipo de elemento de GUI 424 ainda pode incluir um

alias de GAP 430, um identificador de elemento de GUI 432 e um identificador de tipo de GUI 434.

[00112] A Figura 5 mostra um GAP 500 e uma interface de usuário de exemplo 502 para a lógica de tipificação 316. O operador seleciona os elementos de GUI usando, por exemplo, o cursor do mouse. Quando selecionada, a lógica de tipificação 316 destacada o elemento de GUI selecionado. No exemplo mostrado na Figura 4, o elemento de GUI selecionado é a caixa de texto 504.

[00113] Em resposta a seleção de um elemento de GUI, a lógica de tipificação 316 exibe um requisitante de tipo 506. O requisitante de tipo 506 provê uma lista suspensa 508 a qual lista os tipos de elemento de GUI disponíveis definidos no modelo de tipo de GUI de referência 122. O operador seleciona um tipo de elemento para atribuição ao elemento de GUI selecionado 504, neste caso, "US-StateTextBox". Clicar no botão 'OK' dirige a lógica de tipificação 316 para a criação do mapeamento do tipo US-StateTextBox para a caixa de texto 504. Clicar em 'Cancel' dirige a lógica de tipificação 316 para não tomar nenhuma medida.

[00114] A Figura 6 mostra um fluxograma 600 do processamento realizado pela lógica de tipificação 316. A lógica de tipificação 316 monitora quanto à entrada de operador (602). Em particular, a lógica de tipificação 316 usa a entrada de operador para determinar um modelo de tipo de GUI de referência selecionado e o GAP que o operador tipificará (604). Por exemplo, o operador pode enviar um modelo de tipo de GUI de referência aplicável a partir de uma lista de modelos de tipo de GUI de referência, e pode especificar o programa executável concretizando o GAP a partir de uma lista de GAPs conhecidos.

[00115] A lógica de recuperação de tipo 320 recupera uma lista de seleção de tipo de elemento de GUI (606) a partir do modelo de tipo de

GUI de referência selecionado. Continuando o exemplo dado acima na Tabela 1, a lógica de recuperação de tipo 320 recupera a lista {SSNEntryBox, SSNDisplayBox, StaticWindowLabel, 3DButton, HelpTextEntryBox, MenuWidget, e US-StateTextBox} a partir do modelo de tipo de GUI de referência. A lista contém os tipos de elemento de GUI permissíveis que o operador pode atribuir a qualquer dado elemento de GUI.

[00116] A lógica de tipificação 316 também pode exibir o GAP selecionado (608). Em uma implementação, a lógica de tipificação 316 inicia a execução do GAP. A lógica de seleção de elemento 318, então, monitora quanto a uma entrada de operador (610). Em particular, a lógica de seleção de elemento 318 monitora quanto a cliques de mouse, entrada de teclado ou outra entrada para a determinação de um elemento de GUI selecionado (612). A lógica de seleção de elemento 318 destaca o elemento de GUI selecionado (614). Como exemplos, a lógica de seleção de elemento 318 pode desenhar uma borda em torno do elemento, mudar a cor do elemento, fazer piscar o elemento ou de outra forma destacar o elemento de GUI selecionado.

[00117] A lógica de seleção de tipo 322 exibe a lista de seleção de tipo de elemento de GUI (616). Por exemplo, a lógica de seleção de tipo 322 pode exibir o requisitante de tipo 506. A lógica de seleção de tipo 322 monitora quanto a uma entrada de operador (618), tal como uma seleção de lista suspensa de um tipo de elemento a partir da lista de seleção de tipo de elemento de GUI. Em particular, a lógica de seleção de tipo 322 obtém uma seleção de tipo de elemento de GUI que especifica o tipo de elemento de GUI selecionado a partir da lista de seleção de tipo de elemento de GUI exibida (620).

[00118] Se o operador aceitar a seleção, a lógica de associação de tipo 324 criará um mapeamento de tipo de elemento de GUI (622).

Especificamente, a lógica de associação de tipo 324 cria um mapeamento que liga o tipo de elemento de GUI selecionado ao elemento de GUI selecionado. Para essa finalidade, a lógica de associação de tipo 324 pode criar uma entrada de mapeamento de tipo no modelo de tipo de GUI correspondente ao GAP selecionado no depósito 124 que especifica um campo de alias de GAP, um identificador de elemento de GUI 404 para o elemento de GUI selecionado e um identificador de tipo de GUI 406 de acordo com o tipo de em selecionado por operador. A Figura 4 proporciona exemplos de mapeamentos de tipo de elemento de GUI.

[00119] Além disso, a lógica de tipificação 316 pode comunicar o mapeamento de tipo de elemento de GUI para sistemas externos. Para essa finalidade, a lógica de criação de mensagem 326 pode construir uma mensagem de especificação de tipo de elemento de GUI (624). A mensagem de especificação de tipo pode incluir o mapeamento, um cabeçalho de mensagem de especificação de tipo e um terminador de mensagem de especificação de tipo. A lógica de criação de mensagem 326 também pode comunicar a mensagem de especificação de tipo de elemento de GUI para o depósito de metadados 202 (626).

[00120] A Figura 7 mostra um exemplo do sistema 102 no qual uma lógica de mapeamento de versão de elemento de GUI ("lógica de mapeamento de versão") 702 reside na memória 114. A lógica de mapeamento de versão 702 pode ser incluída na lógica de tipificação e de mapeamento de GUI 120, mas também pode ser um módulo separado ou implementado de outras formas. A lógica de mapeamento de versão 702 pode incluir uma lógica de seleção de elemento 704, uma lógica de criação de mapeamento 706 e uma lógica de criação de mensagem 708. A lógica de mapeamento de versão 702 gera mapeamentos de versão de elemento de GUI (por exemplo, o

mapeamento de versão 710).

[00121] A Figura 8 mostra um exemplo do mapeamento de versão de elemento de GUI ("mapeamento de versão") 802. O mapeamento de versão 802 inclui um alias de GAP de origem 804, um identificador de elemento de GUI de origem 806, um alias de GAP de destino 808 e um identificador de elemento de GUI de destino 810. Campos adicionais, em menor quantidade ou diferentes podem ser incluídos no mapeamento de versão 802.

[00122] O alias de GAP de origem 804 especifica um identificador para um GAP ("o GAP de origem") que inclui um primeiro elemento de GUI selecionado, enquanto o alias de GAP de destino 808 especifica um identificador para um GAP (o "GAP de destino") que inclui um segundo elemento de GUI selecionado que deve ser ligado ao primeiro elemento de GUI selecionado. Os aliases de GAP 804 e 808 podem ser identificadores únicos que distinguem entre GAPs, tais como identificadores que diferenciam a versão de GAP atual e a versão de GAP subsequente. O identificador de elemento de GUI de origem 806 provê um identificador único para o elemento de GUI selecionado no GAP de origem, enquanto o identificador de elemento de GUI de destino 810 provê um identificador único para o elemento de GUI selecionado no GAP de destino.

[00123] A Figura 8 também mostra um exemplo específico de um mapeamento de versão 812. O mapeamento de elemento 812 especifica um alias de GAP de origem 814 de "University Directory1", significando a versão subsequente de um GAP de diretório de universidade. O elemento de GUI de origem sendo mapeado (por exemplo, uma caixa de combinação), tem o identificador de elemento único 816 "0x30fc8" etiquetado pelo rótulo "HWND". O mapeamento de elemento 812 também especifica um alias de GAP de destino 818 de "University Directory0", significando a versão atual de um GAP de

diretório de universidade. O elemento de GUI de destino sendo mapeado (por exemplo, uma caixa de lista suspensa) tem o identificador de elemento único 820 "0x30fc0" etiquetado pelo rótulo "HWND". Assim, o mapeamento de versão 812 estabelece que uma caixa de lista suspensa em particular na versão subsequente do GAP corresponde a uma caixa de combinação em particular na versão de GAP atual.

[00124] A Figura 8 também mostra um exemplo de uma mensagem de especificação de mapeamento de elemento de GUI 822. A mensagem de especificação de mapeamento de elemento de GUI 822 pode incluir um cabeçalho de mensagem de especificação de mapeamento de elemento de GUI 824 e um terminador de mensagem de especificação de mapeamento de elemento de GUI 826. O cabeçalho 824 e o terminador 826 significam que os dados na mensagem especificam um mapeamento de elemento entre elementos de GUI em GAPs diferentes. Para essa finalidade, a mensagem de especificação de mapeamento de elemento de GUI 822 pode incluir, ainda, um alias de GAP de origem 828, um identificador de elemento de GUI de origem 830, um alias de GAP de destino 832 e um identificador de elemento de GUI de destino 834.

[00125] Uma extensão opcional para o mapeamento de versão de elemento de GUI 802 é o campo de nível de confiança 811. O campo de nível de confiança 811 pode especificar um grau de confiabilidade para o mapeamento de versão de elemento de GUI. Quando o mapeamento de versão surge a partir dos esforços de um operador humano, por exemplo, a lógica de mapeamento 702 pode regular o nível de confiança relativamente alto (por exemplo, de 90 a 100%). Quando o mapeamento de versão surge a partir de uma análise automatizada, a lógica de mapeamento de versão 702 pode regular o nível de confiança em um nível especificado (por exemplo, um nível

predefinido para combinação automatizada), ou pode regular um limite que depende da força da análise automatizada.

[00126] Por exemplo, a análise automatizada pode determinar um escore normalizado para qualquer dada tentativa de combinação de um elemento de GUI com um outro elemento de GUI. O campo de nível de confiança 811 então pode especificar o escore normalizado. O campo de nível de confiança 811 ainda pode especificar por que o nível de confiança é regulado para qualquer valor em particular. Além disso, um campo de explicação (por exemplo, um caractere tal como "M" ou "A") pode ser incluído no campo de nível de confiança 811 para denotar que o nível de confiança surge de uma análise Manual ou Automatizada. Um exemplo é o nível de confiança 821, regulado para "88A" para denotar uma análise automatizada e um escore normalizado de 88, enquanto o nível de confiança 836 mostra um valor de confiança de 100, sem especificar uma explicação.

[00127] A Figura 9 mostra a versão de GAP atual 902 e uma versão de GAP subsequente 904, bem como uma interface de usuário de mapeamento de exemplo 906 para a lógica de mapeamento 702. Ambos os GAPs 902 e 904 são mostrados executando e apresentando suas GUIs no visor 121. O operador seleciona elementos de GUI usando, por exemplo, o cursor do mouse, uma entrada de teclado ou outras entradas.

[00128] Mediante uma seleção, a lógica de mapeamento 702 destaca os elementos de GUI selecionados. No exemplo mostrado na Figura 9, o elemento de GUI selecionado na versão de GAP atual 902 é a caixa de lista 908. O elemento de GUI selecionado na versão de GAP subsequente 904 é a lista suspensa 910. Mediante uma seleção de um elemento em cada GAP, a lógica de mapeamento 702 exibe a interface de usuário de mapeamento 906. Clicar no botão 'OK' direciona a lógica de mapeamento 702 para criar o mapeamento de

elemento entre a caixa de lista 908 e a lista suspensa 910. Clicar em 'Cancel' dirige a lógica de mapeamento 702 para não efetuar nenhuma ação.

[00129] A Figura 10 mostra um fluxograma 1000 do processamento realizado pela lógica de mapeamento 702. A lógica de mapeamento 702 monitora quanto a uma entrada de operador (1002) de modo a determinar um primeiro GAP (1004) (por exemplo, a versão de GAP atual) e um segundo GAP (1006) (por exemplo, a versão de GAP subsequente) entre os quais o operador mapeará elementos de GUI individuais. Por exemplo, o operador pode selecionar arquivos de programa executáveis para os GAPs a partir de uma janela de seleção de arquivo para escolher os dois GAPs. A lógica de mapeamento 702 exibe as GUIs para os GAPs selecionados (por exemplo, pela execução dos GAPs) (1008).

[00130] A lógica de seleção de elemento 704 então monitora quanto a uma entrada de operador (1010). Em particular, a lógica de seleção de elemento 704 detecta uma entrada de mouse, de teclado e outros tipos de entrada para determinar quando um operador selecionou elementos de GUI nos GAPs (1012). A lógica de seleção de elemento 704 destaca cada elemento de GUI selecionado (1014). Quando um elemento de GUI a partir de cada GAP tiver sido selecionado, a lógica de seleção de elemento 704 exibe uma interface de usuário de mapeamento de elemento de GUI (1016). Se o operador clicar em 'Cancel', a lógica de mapeamento 702 não precisará efetuar nenhuma ação, mas poderá continuar a assistir quanto a seleções de elemento de GUI adicionais.

[00131] Se o operador clicar em 'OK', a lógica de criação de mapeamento 706 criará um mapeamento de versão de elemento de GUI que especifica que existe uma relação entre os elementos de GUI selecionados (1018). Para essa finalidade, a lógica de criação de

mapeamento 706 pode armazenar um alias de GAP de origem, um identificador de elemento de GUI de origem correspondente ao elemento de GUI selecionado no GAP de origem, um alias de GAP de destino, e um elemento identificador de GUI de destino correspondente ao elemento de GUI selecionado no GAP de destino.

[00132] Adicionalmente, a lógica de criação de mensagem 708 pode construir uma mensagem de especificação de mapeamento de elemento de GUI (1020). Para essa finalidade, a lógica de criação de mensagem 708 pode armazenar um cabeçalho de mensagem de especificação de mapeamento de elemento de GUI e um terminador de mensagem de especificação de mapeamento de elemento de GUI. O cabeçalho e o terminador significam que os dados na mensagem especificam um mapeamento de elemento de GUI entre os elementos de GUI em GAPs diferentes. A mensagem de especificação de tipo de elemento de GUI ainda pode incluir um alias de GAP de origem, um identificador de elemento de GUI de origem, um alias de GAP de destino e um identificador de elemento de GUI de destino. A lógica de criação de mensagem 708 pode então comunicar a mensagem de especificação de mapeamento de elemento de GUI para outros sistemas, tal como um depósito de metadados (1022).

[00133] A Figura 11 mostra uma implementação de exemplo do sistema de tipificação e de mapeamento de elemento de GUI ("sistema") 1100. O sistema 1100 inclui uma lógica de driver 1102 na memória 114, bem como um proxy 1104 que acessa uma tabela de GAP 1106. Na Figura 11, a versão de GAP atual 1108 é mostrada em execução com um gancho 1110, e a versão de GAP subsequente 1112 também é mostrada em execução com um gancho 1114.

[00134] O proxy 1104 inclui uma lógica que insere os ganchos 1110 e 1114 no espaço de processo dos GAPs 1108 e 1112. O proxy 1104 se comunica com os ganchos 1110 e 1114. Em particular, o proxy

1104 pode trocar mensagens com os ganchos 1110 e 1114 para a obtenção do estado de qualquer um ou de todos os elementos de GUI nos GAPs 1108 e 1112. Os ganchos 1110 e 1114 são programas que respondem a mensagens do proxy 1104, e que podem interagir através de uma camada de acessibilidade para descobrir e reportar uma informação sobre os elementos de GUI nos GAPs 1108 e 1112 para o proxy. O sistema operacional geralmente provê a camada de acessibilidade. A camada de acessibilidade expõe uma interface de acessibilidade através da qual o proxy 1104 e os ganchos 1110 e 1114 podem invocar métodos e regular e recuperar valores e características de elemento de GUI e, desse modo, selecionar, destacar, controlar, modificar, atribuir identificadores para ou interagir de outra forma com os elementos de GUI nos GAPs.

[00135] A camada Microsoft (TM) Active Accessibility (MSAA) é um exemplo de uma camada de acessibilidade adequada. Nesse sentido, os GAPs 1108 e 1112 expõem as interfaces de acessibilidade que exportam métodos para acesso e manipulação das propriedades e do comportamento de elementos de GUI. Por exemplo, os GAPs 1108 e 1112 podem empregar a interface IAccessible para se permitirem acesso e controle em relação ao elemento de GUI usando chamadas de API de MSAA. A interface IAccessible ainda facilita aplicativos para exposição de uma árvore de nós de dados que constituem cada janela na interface de usuário com a qual se interage atualmente. A lógica de driver 1112 e o proxy 1104 podem incluir, então, declarações de programa para acesso e controle do elemento de GUI, como se o elemento de GUI fosse um objeto de programação convencional. As chamadas de API de acessibilidade podem incluir: realizar uma ação em um objeto, obter um valor de um objeto, regular um valor no objeto, navegar para o objeto, e regular uma propriedade no objeto, e outras chamadas.

[00136] O proxy 1104 pode ser um programa daemon e pode começar antes da lógica de driver 1102. O proxy 1104 pode estar ciente de um ou mais GAPs. Quando o proxy 1104 começa, ele carrega a tabela de GAP 1106, a qual pode incluir um conjunto predefinido de entradas de GAP quanto às quais o proxy 1104 está ciente. Uma entrada de GAP pode assumir a forma:

[00137] <Alias, <File0, Path0, Dir0, CommandLine0>, <File1, Path1, Dir1, CommandLine1>>

[00138] onde Alias é um nome predefinido único para o GAP (por exemplo, um nome genérico para a versão de GAP atual 1108 e a versão de GAP subsequente 1112), File0 é o nome do programa executável para a versão de GAP atual 1108, Path0 é o caminho absoluto para File0, Dir0 é o caminho absoluto para o diretório a partir do qual File0 deve ser executado, e CommandLine especifica os argumentos de linha de comando para File0. File1, Path1, Dir1 e CommandLine1 provêm parâmetros similares para a versão de GAP subsequente 1112.

[00139] Quando a lógica de driver 1102 começa, ela se conecta ao Proxy de forma local ou remota (por exemplo, através de uma porta de Protocolo de Controle de Transmissão (TCP)). Uma vez conectada, a lógica de driver 1102 requisita a tabela de GAP 1106 ao enviar uma mensagem de requisição de tabela de GAP para o proxy 1104. O proxy 1104 responde pelo envio de uma mensagem de resposta de tabela de GAP incluindo a tabela de GAP 1106 para a lógica de driver 1102. Uma troca de mensagem de exemplo é mostrada na Tabela 2:

Tabela 2
mensagem de requisição de tabela de GAP <GetGapTable/>
mensagem de resposta de tabela de GAP
<GapTable> <GAP Alias = "name" <V_N File="gap.exe" Path="c:\path\N" CommandLine="-c1"/> <V_N1 File="gap.exe" Path="c:\path\N1" CommandLine="-c1"/> </GAP> </GapTable>

[00140] A lógica de driver 1102 então pode prover uma lista de GAPs para escolher a partir do operador quanto a tipificar o GAP (Figura 5, (602), (604)) ou realizar um mapeamento de elemento de GUI (Figura 10, (1002), (1004), (1006)). A lógica de driver 1102 então pode criar uma mensagem de carregar GAP, por exemplo, <LoadGap Alias="name"/> e enviar a mensagem de carregar GAP para o proxy 1104 para se começar qualquer GAP selecionado (o qual então exibirá sua interface de usuário) (Figuras 6 e 10, (608), (1008)). Quando o operador está realizando um mapeamento de elemento, uma mensagem de carregar GAP pode fazer com que o proxy 1104 comece múltiplas versões de um identificador de GAP em conjunto com a tabela de GAP 1106 na seção de <GAP>.

[00141] Após começar os GAPs, o proxy 1104 injeta ganchos no espaço de processo de GAPs. O gancho se conecta ao proxy 1104 e envia uma mensagem de confirmação (por exemplo, <GAP File="gap.exe" Instance="192"/>). O proxy 1104 envia uma mensagem de sucesso (por exemplo, <Loaded Alias="name" VN="192" VN1="193"/>) para a lógica de driver 1102, desse modo reconhecendo que os GAPs foram começados de forma bem-sucedida.

[00142] O operador pode requisitar o estado atual de cada GAP começado a partir da lógica de driver 1102. Em resposta, a lógica de driver 1102 envia uma mensagem de requisição de estado (por

exemplo, <GetState Alias="name"/>) para o proxy 1104. Por sua vez, o proxy 1104 localiza a conexão com os ganchos correspondentes dos GAPs e envia uma mensagem de requisição de estado (por exemplo, <GetState/>) para os ganchos. Os ganchos criam um estado de GAP (incluindo identificadores únicos para os elementos de GUI), tal como uma árvore de estado, codifica-o (por exemplo, em um formato em XML), e o envia para o proxy 1104. O proxy 1104 encaminha o estado de GAP para a lógica de driver 1102. Uma mensagem de estado de GAP de exemplo enviada pelo proxy 1104 é mostrada na Tabela 3.

Tabela 3

mensagem de estado de GAP

```
<State SeqNumber="1" Name="name" Alias="name" ProcessID="972">
        <GUIElement Alias="name">
                <Location x="15" y="200" width="23" height="98"/>
                <Description>Action</ Description>
                <DefAction>Action</DefAction>
                <UniqueID>0xcafebabe</UniqueID>
                <Class>LISTBOX</Class>
                <Values>
                        <Value SeqNumber="1">someval</Value>
                        ……
                </Values>
        </GUIElement>
        ……
</State>
```

[00143] O estado de GAP contém uma informação sobre os elementos de GUI compondo uma dada tela, bem como os valores destes elementos e seus identificadores atribuídos. O estado de GAP especifica os elementos de GUI de GAP e os valores dos elementos de GUI. Em uma implementação, o estado de GAP é refletido em uma estrutura de linguagem de marcação extensível (XML), onde o elemento 'State' tem um ou mais elementos filhos 'GAP', cujos elementos filhos por sua vez são 'GUIElements'. Por exemplo, os elementos de GUI podem ser containeres ou básicos. Os elementos de GUI de container contêm outros elementos, enquanto os elementos

básicos não contêm outros elementos. A estrutura de XML reflete a hierarquia de contenção ao permitir que os GUIElements contenham outros GUIElements.

[00144] Na estrutura de XML, o atributo SeqNumber pode designar um número de sequência único do estado dentro do GAP. Uma vez que os estados são mapeados para telas de GUI, a cada estado pode ser dado um nome o qual é especificado pelo atributo 'Name'. Os atributos Alias e ProcessID podem denotar o alias do GAP e seu identificador de processo de instância, respectivamente. O identificador de processo de instância pode diferenciar entre a versão de GAP atual e a versão de GAP subsequente.

[00145] A lógica de tipificação e de mapeamento 120 pode aceitar uma entrada de operador (por exemplo, uma entrada de mouse) através da qual o operador identifica um objeto de GUI ao apontar o cursor para ele (Figuras 10 e 6, (1012) e (612)). A lógica de tipificação e de mapeamento 120 então pode desenhar uma moldura em torno do elemento (Figuras 10 e 6 (1014) e (614)). A lógica de tipificação e de mapeamento 120 então pode exibir uma lista de seleção de tipo de elemento de GUI (Figura 6, (616)) ou uma interface de usuário de mapeamento de elemento de GUI (Figura 10, (1016)). O operador seleciona o tipo apropriado para o objeto (Figura 6, (620)) ou verifica se um mapeamento deve ser criado entre dois objetos selecionados. Em qualquer caso, o proxy 1104 envia um mapeamento de tipo de elemento de GUI ou um mapeamento de versão de elemento de GUI para a lógica de driver 1102. Por sua vez, a lógica de driver 1102 pode armazenar os mapeamentos no depósito 116, e pode criar e comunicar uma mensagem de especificação de elemento de GUI ou uma mensagem de especificação de tipo de elemento de GUI para o depósito de metadados 202 (Figura 6, (626) e Figura 10, (1022)).

[00146] A Figura 11 também mostra um depósito de modelo de tipo

de referência 1116. O sistema 1110 pode tirar o modelo de tipo de GUI de referência 122 atualmente sendo aplicado a partir de um de muitos modelos de tipo de referência disponíveis no depósito de modelo de tipo de referência 1116. Os exemplos mostrados na Figura 11 de modelos de tipo de referência incluem um modelo de tipo de referência de jogo de interpretação de papéis (role playing game - RPG) online 1118, que pode ser selecionado pelo operador, quando do projeto ou da tipificação de um GAP que implementa uma funcionalidade de jogo online, e um modelo de tipo de referência de registros de paciente 1120 que o operador pode selecionar quando projetando ou tipificando um GAP de assistência médica. Os exemplos adicionais incluem o modelo de tipo de referência de processador de texto 1122 que o operador pode selecionar quando tipificando ou construindo um GAP de processador de texto, e um modelo de tipo de referência de envio de mensagem instantânea (IM) 1124, que o operador pode selecionar quando projetando ou tipificando um GAP de IM.

[00147] A Figura 12 mostra uma modalidade de uma arquitetura de ferramenta de evolução de metadados 1200. A arquitetura 1200 inclui uma interface 136, um processador 129, uma memória 130, e um depósito de metadados 138. A arquitetura 1200 pode se comunicar com outros sistemas através da interface 136. Por exemplo, a arquitetura 1200 pode receber requisições de informação de metadados e enviar respostas àquelas requisições através da interface 136. De forma alternativa ou adicional, a arquitetura 1200 pode enviar instruções para a interface 136 para exibição de um alerta em um terminal não local e pode receber respostas ao alerta através da interface 136. De forma alternativa ou adicional, o processador 129 pode enviar instruções para exibição de um alerta em um visor 1236 e pode receber respostas ao aleta. O processador 129 executa a lógica armazenada na memória 130. O depósito de metadados 138 recebe,

recupera e armazena uma informação de metadados processada pelo processador 129.

[00148] A memória 130 pode incluir uma lógica de processamento de metadados 132, uma lógica de gerenciamento de banco de dados e de arquivo 134 e mensagens 1214. A lógica de processamento de metadados 132 pode instruir o processador 129 para realizar fluxos de processo para manutenção e migração de metadados. A lógica de gerenciamento de banco de dados e de arquivo 134 pode instruir o processador 129 para realizar processos relevantes para armazenamento de dados, recuperação e manipulação para e a partir do depósito de metadados 138. As mensagens 1214 podem ser armazenadas na memória, quando recebidas a partir da interface 136 e manipuladas pelo processador 129 de acordo com instruções a partir da lógica de processamento de metadados 132.

[00149] A lógica de processamento de metadados 132 inclui uma lógica de manipulação de mensagem de metadados 1216, uma lógica de processamento de tipo 1218, uma lógica de processamento de mapeamento 1220, uma lógica de processamento de notação 1222 e uma lógica de migração de metadados 1224. A lógica de manipulação de mensagem de metadados 1216 pode instruir o processador 129 para armazenar as mensagens 1214 recebidas a partir da interface 136 na memória 130 e processar as mensagens 1214 conforme descrito abaixo. A lógica de manipulação de mensagem de metadados 1216 pode incluir uma lógica de comunicação 1226 e uma lógica de análise gramatical 1228. A lógica de comunicação 1226 pode instruir o processador 129 para enviar e receber mensagens 1214 através da interface 136. A lógica de comunicação 1226 também pode instruir o processador 129 para armazenar as mensagens 1214 na memória 130. De forma alternativa ou adicional, a lógica de comunicação 1226 pode enviar instruções para a interface 136 para a exibição de um

alerta para um terminal não local e pode instruir o processador 129 para processar instruções recebidas pela interface 136, em resposta ao alerta. De forma alternativa ou adicional, a lógica de comunicação 1226 pode instruir o processador 129 para enviar instruções para o visor 1236 para exibição de um alerta, e o processador 129 pode processar instruções recebidas em resposta ao alerta. A lógica de análise gramatical 1228 pode instruir o processador 129 para analisar gramaticalmente as mensagens 1214. Por exemplo, a lógica de análise gramatical 1228 pode instruir o processador 129 para extrair uma informação de identificação de metadados a partir das mensagens.

[00150] A lógica de processamento de tipo 1218, a lógica de processamento de mapeamento 1220 e a lógica de processamento de notação 1222 podem instruir o processador 129 para o processamento de mensagens de metadados, tais como a mensagem de especificação de tipo 1230, a mensagem de especificação de mapeamento 1232 e a mensagem de notação 1234. Por exemplo, a lógica de processamento de tipo 1218, a lógica de processamento de mapeamento 1220 ou a lógica de processamento de notação 1222 pode instruir o processador 129 para manter um registro de metadados armazenado no depósito de metadados 138. Nesse sentido, o processador 129 pode dirigir uma leitura, uma escrita, um armazenamento, uma cópia, uma atualização, um movimento, um apagamento, uma sobrescrita, uma anexação em apenso, ou manipular de outra forma dados armazenados em um registro de metadados no depósito de metadados 138. A lógica de processamento de tipo 1218, a lógica de processamento de mapeamento 1220 ou a lógica de processamento de notação 1222 pode ser realizada em conjunto com processos a partir da lógica de gerenciamento de banco de dados e de arquivo 134, As mensagens 1214 incluem como

exemplos mensagens de especificação de tipo 1230, mensagens de especificação de mapeamento 1232 e mensagens de notação 1234, mas podem incluir outras mensagens relacionadas a metadados. As mensagens de especificação de tipo 1230, as mensagens de especificação de mapeamento 1232 e as mensagens de notação 1234 são discutidas em maiores detalhes com respeito às Figuras 4, 8 e 14, respectivamente.

[00151] A Figura 13 mostra uma implementação do depósito de metadados 138. O depósito de metadados 138 pode ser organizado de muitas formas diferentes, contudo. O depósito de metadados 138 pode incluir um registro de metadados de GAP 0 1302, um registro de metadados de GAP 1 1304 até um registro de metadados de GAP 'j' 1306, e um registro de mapeamento de versão de elemento de GUI 1308. Os registros de metadados de GAP 1302, 1304 e 1306 podem armazenar metadados associados a um GAP específico ou a uma versão de GAP. O registro de mapeamento de versão de elemento de GUI 1308 pode armazenar mapeamentos a partir de um elemento de GUI para um outro elemento de GUI.

[00152] Cada registro de metadados de GAP pode incluir um identificador de GAP. Os identificadores de GAP 1310, 1312, e 1314 podem servir para a identificação de um GAP, de uma versão de um GAP ou de ambos. Por exemplo, o registro de metadados de GAP 0 1302 contém um identificador de GAP 1 de "University Directory0" e o registro de metadados de GAP 1 1304 contém um identificador de GAP 1 1312 de "University Directory1". Neste caso, "University Directory0" pode servir para a identificação do GAP inteiro como "University Directory0". De forma alternativa ou adicional, "University Directory0" pode servir para a identificação do GAP como a versão 0 (por exemplo, a versão atual) do GAP "University Directory". O depósito de metadados 138 pode armazenar registros de metadados

para múltiplos GAPs, bem como múltiplos registros de metadados para cada uma das múltiplas versões de cada GAP.

[00153] O registro de metadados de GAP 0 1302 adicionalmente pode incluir o registro de metadados de elemento de GUI 1 1316 através do registro de metadados de elemento de GUI 'n' 1318. O número total 'n' de registros de metadados de elemento de GUI armazenados em cada registro de metadados de GAP pode variar, dependendo da complexidade do GAP. Cada registro de metadados de elemento de GUI pode corresponder a um elemento de GUI em um GAP ou uma versão de um GAP, e cada elemento de GUI em um gpa pode ter um registro de metadados de elemento de GUI correspondente a um registro de metadados de GAP. Por exemplo, o registro de metadados de GAP 0 1302 contém o registro de metadados de elemento de GUI 'n' 1318 indicando que o GAP 0 pode ser composto por 'n' ou mais elementos de GUI identificáveis. De forma alternativa ou adicional, o registro de metadados de elemento de GUI 'n' 1318 pode indicar que o registro de metadados de GAP 0 1302 atualmente contém 'n' registros de metadados de elemento de GUI, onde 'n' pode ser um valor inteiro de 0 até o número total de elementos de GUI no GAP 9. Todo elemento de GUI em um GAP pode não ter um registro de metadados de elemento de GUI. De modo similar, o registro de metadados de GAP 1 1304 pode conter o registro de metadados de elemento de GUI 'k', e um registro de metadados de GAP 'j' pode conter o registro de metadados de elemento de GUI 'm' 1322.

[00154] Cada um dos registros de metadados de elemento de GUI 1316, 1318, 1320 e 1322 pode incluir um identificador de elemento de GUI 1324, um identificador de tipo 1326, uma notação 1328, um mapeamento de elemento de GUI 1330 e outros metadados 1332. Um identificador de elemento de GUI 1324 pode servir para a identificação

de um elemento de GUI no GAP ou na versão de GAP, usando-se um número único, uma string de caracteres ou outros índices. Por exemplo, um ID de elemento pode ser "0x30fb0". Um identificador de tipo 1326 pode ser uma classificação de um elemento de GUI que defina semântica de nível alto, anotações e propriedades, comportamentos permitidos ou restritos, e valores associados ao elemento de GUI. Por exemplo, um identificador de tipo pode ser "US-StateTextBox", o que pode especificar um tipo de elemento de GUI de caixa de texto que apenas aceita strings correspondentes aos nomes dos estados dos Estados Unidos. Esta informação pode vir a partir do conhecimento de um engenheiro de testes quando ele ou ela tentar entender a semântica de cada elemento de GUI no GAP sob teste.

[00155] Uma notação 1328 pode incluir texto, notas, comentários informais, restrições, ou uma informação derivada de uma especificação técnica que um programador pode desejar levar para um outro programador sobre um elemento de GUI em particular. Por exemplo, uma notação 1328 pode incluir o texto "State Names Only" como um método informal de levar para um outro programador que apenas strings correspondentes aos nomes dos estados dos Estados Unidos devem ser incluídas. Um mapeamento de elemento de GUI 1330 pode identificar um elemento de GUI em um outro GAP ou versão de GAP correspondente ao elemento de GUI associado ao registro de metadados de elemento de GUI. Por exemplo, um mapeamento de elemento de GUI 1330 pode incluir os valores "University Directory1" e "0x30fc8" para indicar que o elemento de GUI associado a este registro de metadados de elemento de GUI corresponde ao elemento de GUI 0x30fc8 no GAP University Directory, versão 1. Adicionalmente, outros metadados 1332 podem ser armazenados em associação com um registro de metadados de elemento de GUI.

[00156] O depósito de metadados 138 pode incluir qualquer número de registros de mapeamento de versão de elemento de GUI, tal como o registro 1308. O número de registros de mapeamento de versão de elemento de GUI 1308 pode variar, dependendo do número de GAPs ou de versões de GAP. Por exemplo, cada registro de mapeamento de versão de elemento de GUI 1308 pode incluir mapeamentos a partir de uma versão de GAP específica para uma outra versão de GAP específica. De forma alternativa ou adicional, cada registro de mapeamento de versão de elemento de GUI 1308 pode incluir todos os mapeamentos entre todas as versões de um único GAP. O exemplo da Figura 13 mostra que o registro de mapeamento de versão de elemento de GUI 1308 inclui mapeamentos de versão de elemento de GUI 1334 e 1336. O número de mapeamentos de versão de elemento de GUI 1334 e 1336 em um registro de mapeamento de versão de elemento de GUI 1308 pode variar de acordo com o número de mapeamentos feitos entre elementos de GUI de GAPs ou de versões de GAP diferentes.

[00157] Cada mapeamento de versão de elemento de GUI 1334 ou 1336 pode incluir um identificador de alias de GAP de origem 1338, um identificador de elemento de GUI de origem 1340, um identificador de alias de GAP de destino 1342, um identificador de elemento de GUI de destino 1344 e um valor de nível de confiança 1346. O mapeamento de versão de elemento de GUI 1348 provê um exemplo de um mapeamento de versão de elemento de GUI usando linguagem de marcação extensível (XML). O mapeamento de versão de elemento de GUI 1348 inclui um identificador de alias de GAP de origem 1350, um identificador de elemento de GUI de origem 1352, um identificador de alias de GAP de destino 1354, um identificador de elemento de GUI de destino 1356 e um valor de nível de confiança 1358. Neste exemplo, o mapeamento indica uma correspondência entre o elemento

de GUI 0x30fc8 de GAP University Directory, versão 1 (por exemplo, uma versão subsequente) com o elemento de GUI 0x80fc0 de GAP University Directory, versão 0 (por exemplo, uma versão atual), com um nível de confiança de 1200. Os valores de nível de confiança podem usar valores decimais entre 0 e 1, valores inteiros entre 0 e 100, ou qualquer outra escala, para indicarem a intensidade da certeza que se tem quanto ao mapeamento entre os elementos de GUI ser um mapeamento correto. Por exemplo, um mapeamento provido por um usuário humano pode ter uma confiança alta ou absoluta de 1 ou 1200 por cento, onde um mapeamento provido por um programa de avaliação de mapeamento usando uma comparação de propriedade de elemento de GUI pode ter uma confiança mais baixa ou nenhuma. De forma alternativa ou adicional, um mapeamento provido por um programa de avaliação de mapeamento usando uma comparação de propriedade de elemento de GUI pode ter uma confiança alta ou absoluta, onde um mapeamento provido por um usuário humano pode ter uma confiança mais baixa ou nenhuma.

[00158] O processador 129 usa a lógica de gerenciamento de banco de dados e de arquivo 134 para acessar e manipular qualquer um dos registros de metadados de GAP 1302, 1304 ou 1306, os registros de metadados de elemento de GUI 1316, 1318, 1320 ou 1322, e/ou registros de mapeamento de versão de elemento de GUI armazenados no depósito de metadados 138. O acesso e a manipulação dos dados no depósito de metadados 138 podem incluir a leitura, a escrita, o armazenamento, a cópia, a atualização, o movimento, o apagamento, a sobrescrita, a anexação em apenso, ou qualquer outra função realizada nos dados.

[00159] A Figura 14 mostra um exemplo de um mapeamento de notação de elemento de GUI 1400 que pode ser um componente de uma mensagem de notação de elemento de GUI 1234. O formato de

mapeamento de notação 1400 inclui um alias de GAP 1402, um identificador de elemento de GUI 1404 e uma notação de GUI 1406. Campos adicionais, a menos ou diferentes podem ser incluídos no mapeamento de notação 1400.

[00160] O alias de GAP 1402 especifica um identificador para o GAP, o qual inclui o elemento de GUI ao qual uma notação está sendo aplicada. O alias de GAP 1402 pode ser um identificador único que distingue entre GAPs ou versões de GAP, incluindo uma versão de GAP atual e uma versão subsequente do mesmo GAP. O identificador de elemento de GUI 1404 provê um identificador único para o elemento de GUI o qual está sendo anotado. A notação de GUI 1406 especifica a notação sendo atribuída ao elemento de GUI (por exemplo, o texto "State names only").

[00161] A Figura 14 também mostra dois exemplos de mapeamentos de notação de GUI 1408 e 1410 usando uma representação em XML. O mapeamento de notação 1408 é um mapeamento para um elemento de GUI de Window. O alias de GAP 1412 é "University Directory0", significando a versão atual de um GAP de diretório de universidade. O elemento de GUI sendo anotado tem um identificador de elemento único 1414 "0x30fb0" anotado pelo identificador de HWND e estabelecido, por exemplo, por uma interface de camada de acessibilidade. A notação de GUI 1416 para o elemento de GUI de Window é o texto "State names only".

[00162] O mapeamento de notação 1410 é um mapeamento para um elemento de GUI de Menu Item. O alias de GAP 1418 é "University Directory1", significando a versão subsequente do GAP de diretório de universidade. O elemento de GUI sendo anotado tem o identificador de elemento único 1420 "OpenFile" conforme especificado pelo campo Name. A notação de GUI 1422 para o elemento de GUI de Window é o texto "Opens to Default Directory", conforme especificado pelo campo

Annotation.

[00163] A Figura 14 também mostra um exemplo de uma mensagem de notação de elemento de GUI 1424 em uma representação em XML. A mensagem de notação de elemento de GUI 1424 pode incluir um cabeçalho de mensagem de notação de elemento de GUI 1426 ("NotationGuiObject") e um terminador de mensagem de notação de elemento de GUI 1428 ("/NotationGuiObject"). O cabeçalho 1426 e o terminador 1428 significam que os dados dentro da mensagem especificam uma notação para um elemento de GUI. Para essa finalidade, a mensagem de notação de elemento de GUI 1424 ainda pode incluir um alias de GAP 1430, um identificador de elemento de GUI 1432 e uma notação de GUI 1434.

[00164] A Figura 15 mostra um fluxograma 1500 de processamento de mensagem de metadados que a lógica de processamento de metadados 132 pode realizar. A lógica de processamento de metadados 132 pode obter uma mensagem de metadados (1502). Por exemplo, a mensagem de metadados pode ser obtida através da interface 136 e armazenada na memória 130. A lógica de processamento de metadados 132 pode instruir o processador 129 para, então, analisar gramaticalmente a mensagem de metadados (1504). Por exemplo, a lógica de processamento de metadados 132 pode analisar gramaticalmente uma mensagem de metadados para a obtenção de uma informação de identificação de elemento, uma informação de identificação de GAP, dados de notação e outros metadados. A lógica de processamento de metadados 132 então pode manter registros de metadados com base na informação extraída a partir da mensagem de metadados analisada gramaticalmente (1506). A manutenção de registros de metadados pode incluir a leitura, a escrita, o armazenamento, a cópia, a atualização, o movimento, o

apagamento, a sobrescrita, a anexação em apenso ou a manipulação de outra forma dos dados dentro dos registros de metadados. A lógica de processamento de metadados 132 então pode checar se quaisquer mensagens a mais estão disponíveis para processamento (1508). Se mais mensagens estiverem disponíveis, a lógica de processamento de metadados 132 então pode circular de volta e obter a próxima mensagem de metadados (1502). Se não houver mais mensagens disponíveis, então, a lógica de processamento de metadados 132 poderá terminar.

[00165] A Figura 1600 mostra um fluxograma 700 de processamento de tipo que pode ser realizado pela lógica de processamento de tipo 1218. A lógica de processamento de tipo 1218 primeiramente pode obter uma mensagem de especificação de tipo (1602). A mensagem de especificação de tipo pode estar no formato de exemplo mostrado para a mensagem de especificação de tipo de elemento de GUI 424. A lógica de processamento de tipo 1218 então pode extrair um alias de GAP a partir da mensagem de especificação de tipo (1604). O alias de GAP pode ser delimitado em uma declaração de XML, conforme ilustrado pelo alias de GAP 430. A lógica de processamento de tipo 1218 então pode extrair um identificador de elemento de GUI a partir da mensagem de especificação de tipo (1606). O identificador de elemento de GUI pode ser delimitado em uma declaração de XML, conforme ilustrado pelo identificador de elemento de GUI 432. A lógica de processamento de tipo 1218 então pode extrair um identificador de tipo de GUI a partir da mensagem de especificação de tipo (1608). O identificador de tipo de GUI pode ser delimitado em uma declaração em XML, conforme ilustrado pelo identificador de tipo de GUI 434.

[00166] A lógica de processamento de tipo 1218 então pode determinar se um registro de metadados de tipo correspondente ao

alias de GAP e ao identificador de elemento de GUI já existe (1610). Se um registro de metadados de tipo já existir para o alias de GAP e para o identificador de elemento de GUI, então, a lógica de processamento de tipo 1218 armazenará o identificador de tipo de GUI no registro de metadados de tipo (1612). A lógica de processamento de tipo 1218 pode armazenar o identificador de tipo de GUI ao sobrescrever um identificador de tipo de GUI já existente ou armazenando o identificador de tipo de GUI em um campo de identificador de tipo de GUI em branco. Além disso, a lógica de processamento de tipo 1218 pode exibir um alerta de requisição de confirmação, antes de sobrescrever um identificador existente, ou pode empregar qualquer outra técnica de resolução de conflito antes de armazenar ou sobrescrever dados. Se um registro de metadados de tipo não existir ainda para o alias de GAP e o identificador de elemento de GUI, então, a lógica de processamento de tipo 1218 poderá criar um registro de metadados de tipo para o alias de GAP e o identificador de elemento de GUI (1614). A lógica de processamento de tipo 1218 então pode armazenar o identificador de tipo de GUI no registro de metadados de tipo (1612).

[00167] A Figura 17 mostra uma primeira parte de um fluxograma 800 de processamento de mapeamento que pode ser realizado pela lógica de processamento de mapeamento 1220. A lógica de processamento de mapeamento 1220 primeiramente pode obter uma mensagem de especificação de mapeamento (1702). A mensagem de especificação de mapeamento pode ser no formato de exemplo mostrado para a mensagem de mapeamento de versão de elemento de GUI 822. A lógica de processamento de mapeamento 1220 então pode extrair um alias de GAP de origem a partir da mensagem de especificação de mapeamento (1704). O alias de GAP de origem pode ser delimitado em uma declaração de XML, conforme ilustrado pelo

alias de GAP de origem 828. A lógica de processamento de mapeamento 1220 então pode extrair um alias de GAP de destino a partir da mensagem de especificação de mapeamento (1706). O alias de GAP de destino pode ser delimitado em uma declaração em XML, conforme ilustrado pelo alias de GAP de destino 832. A lógica de processamento de mapeamento 1220 então pode extrair um identificador de elemento de GUI de origem a partir da mensagem de especificação de mapeamento (1708). O identificador de elemento de GUI de origem pode ser delimitado em uma declaração em XML, conforme ilustrado pelo identificador de elemento de GUI de origem 830. A lógica de processamento de mapeamento 1220 então pode extrair um identificador de elemento de GUI de destino a partir da mensagem de especificação de mapeamento (1710). O identificador de elemento de GUI de destino pode ser delimitado em uma declaração de XML, conforme ilustrado pelo identificador de elemento de GUI de destino 834. A lógica de processamento de mapeamento 1220 então pode extrair um valor de confiança a partir da mensagem de especificação de mapeamento (1712). O valor de confiança pode ser delimitado em uma declaração em XML, conforme ilustrado pelo valor de confiança 836.

[00168] A Figura 18 mostra uma segunda parte de um fluxograma 18000 de processamento de mapeamento que pode ser realizado pela lógica de processamento de mapeamento 1220. A lógica de processamento de mapeamento 1220 pode decidir qual registro de metadados de mapeamento atualizar (1802). A decisão pode ser com base em regulagens pré-existentes ou padronizadas. De forma alternativa ou adicional, a decisão pode ser com base em uma instrução de mapeamento recebida a partir de um usuário, em resposta a um alerta de instrução. Se a instrução de mapeamento especificar uma atualização do registro de mapeamento de versão de

elemento de GUI, então, a lógica de processamento de mapeamento 1220 poderá criar um mapeamento de versão de GUI (1804). O mapeamento de versão de GUI pode ser delimitado em uma declaração de XML, conforme ilustrado pelo mapeamento de versão 812. A lógica de processamento de mapeamento 1220 então pode armazenar o mapeamento de versão de GUI (1806). Por exemplo, o mapeamento de versão de GUI pode ser armazenado no registro de mapeamento de versão de elemento de GUI 1308.

[00169] Se a instrução de mapeamento especificar uma atualização dos registros de metadados de elemento de GUI, então, a lógica de processamento de mapeamento 1220 poderá localizar um registro de metadados de GAP de destino em um depósito de metadados (1808). O depósito de metadados pode ser o depósito de metadados 138. O registro de metadados de GAP de origem pode estar no formato de exemplo mostrado para o registro de metadados de GAP 0 1302. O registro de metadados de GAP de origem pode ser localizado por uma comparação do alias de GAP de fonte extraído a partir da mensagem de especificação de mapeamento (1704) com um identificador de GAP, tal como o identificador de GAP 0 1310. A lógica de processamento de mapeamento 1220 então pode armazenar o identificador de elemento de GUI de destino (1810). Por exemplo, o identificador de elemento de GUI de destino pode ser armazenado no campo de mapeamento de elemento de GUI 1330. De forma alternativa ou adicional, o nível de confiança pode ser armazenado. Por exemplo, o nível de confiança pode ser armazenado no campo de outros metadados 1332. A lógica de processamento de mapeamento 1220 então pode localizar um registro de GUI de destino (1812). O registro de metadados de GAP de destino pode ser similar ao registro de metadados de GAP 1 1304. O registro de metadados de GAP de destino pode ser localizado por uma comparação do alias de GAP de

destino extraído a partir da mensagem de especificação de mapeamento (1706) com um identificador de GAP, tal como o identificador de GAP 1 1312. A lógica de processamento de mapeamento 1220 então pode armazenar o identificador de elemento de GUI de origem (1814). Por exemplo, o identificador de elemento de GUI de origem pode ser armazenado no campo de mapeamento de elemento de GUI 1330. De forma alternativa ou adicional, o nível de confiança pode ser armazenado. Por exemplo, o nível de confiança pode ser armazenado no campo de outros metadados 1332. A lógica de processamento de mapeamento 1330 então pode terminar. Estas etapas não precisam ser realizadas em qualquer ordem em particular. Algumas etapas podem ser adicionadas ou removidas sem se afetarem os objetivos do processo.

[00170] Se a instrução de mapeamento especificar uma atualização dos registros de metadados de elemento de GUI e dos registros de mapeamento de versão de GUI, então, a lógica de processamento de mapeamento 1220 primeiramente poderá localizar um registro de metadados de GAP de origem em um depósito de metadados (1816). A lógica de processamento de mapeamento 1220 então pode armazenar o identificador de elemento de GUI de destino e/ou o nível de confiança (1818). A lógica de processamento de mapeamento 1220 então pode localizar um registro de metadados de GUI de destino (1820). A lógica de processamento de mapeamento 1220 então pode armazenar o identificador de elemento de GUI de origem e/ou o nível de confiança (1822). A lógica de processamento de mapeamento 1220 então pode criar um mapeamento de versão de GUI (1824). A lógica de processamento de mapeamento 1220 então pode armazenar o mapeamento de versão de GUI nos registros de mapeamento de versão de GUI (1826). A lógica de processamento de mapeamento 1220 então pode terminar. Estas etapas não precisam ser realizadas

em qualquer ordem em particular. Algumas etapas podem ser adicionadas ou removidas, sem se afetarem os objetivos do processo.

[00171] A Figura 19 mostra um fluxograma 1900 de processamento de notação que pode ser realizado pela lógica de processamento de notação 1222. A lógica de processamento de notação 1222 primeiramente pode obter uma mensagem de notação (1902). A mensagem de notação pode estar no formato mostrado para a mensagem de notação de elemento de GUI 1424. A lógica de processamento de notação 1222 então pode extrair um alias de GAP a partir da mensagem de notação (1904). O alias de GAP pode ser delimitado em uma declaração de XML, conforme ilustrado pelo alias de GAP 1430. A lógica de processamento de notação 1222 então pode extrair um identificador de elemento de GUI a partir da mensagem de notação (1906). O identificador de elemento de GUI pode ser delimitado em uma declaração de XML, conforme ilustrado pelo identificador de elemento de GUI 1432. A lógica de processamento de notação 1222 então pode extrair uma notação de GUI a partir da mensagem de notação (1908). A notação de GUI pode ser delimitada em uma declaração de XML, conforme ilustrado pela notação de GUI 1434.

[00172] A lógica de processamento de notação 1222 então pode determinar se já existe um registro de metadados de notação correspondente ao alias de GAP e ao identificador de elemento de GUI extraídos a partir da mensagem de notação (1910). Se o registro de metadados de notação já existir para o alias de GAP e o identificador de elemento de GUI, então, a lógica de processamento de notação 1222 poderá armazenar a notação de GUI no registro de metadados de notação (1912), antes da terminação. A lógica de processamento de notação 1222 pode armazenar a notação de GUI ao sobrescrever uma notação de GUI existente, armazenando a notação de GUI em

um campo de notação de GUI em branco, exibindo um alerta antes de sobrescrever uma notação existente, colocando em apenso a notação com uma notação existente, ou usando qualquer outra forma adequada de resolução de conflitos antes do armazenamento de dados. Se um registro de metadados de notação ainda não existir para o alias de GAP e o identificador de elemento de GUI, então, a lógica de processamento de notação 1222 poderá criar um registro de metadados de notação para o alias de GAP e o identificador de elemento de GUI (1914). A lógica de processamento de notação 1222 então pode armazenar a notação de GUI no registro de metadados de notação (1912), antes da terminação.

[00173] A Figura 20 mostra um fluxograma 2000 de processamento de migração de metadados que a lógica de migração de metadados 1224 pode realizar. A lógica de migração de metadados 1224 pode localizar um registro de metadados de origem (2002). A lógica de migração de metadados 1224 então pode localizar um registro de metadados de destino (2004). A lógica de migração de metadados 1224 então pode identificar metadados órfãos (2006). Os metadados órfãos incluem metadados armazenados em um registro de metadados de elemento de GUI em que os metadados também não são armazenados em um outro registro de metadados de elemento de GUI para o qual o primeiro registro de metadados de elemento de GUI é mapeado. De forma alternativa ou adicional, os metadados órfãos podem incluir metadados associados a um elemento de GUI em que o elemento de GUI não tem um mapeamento para um outro elemento de GUI. A lógica de migração de metadados 1224 então pode migrar para quaisquer metadados órfãos (2008). A migração pode ocorrer automaticamente. De forma alternativa ou adicional, a migração pode ocorrer após a exibição de uma mensagem de alerta. Se mais mensagens de origem estiverem disponíveis (2010), a lógica de

migração de metadados 1224 poderá então circular de volta e obter o próprio registro de origem (2002). Se não houver mais registros de origem disponíveis (2010), então, a lógica de migração de metadados 1224 poderá cessar.

[00174] A Figura 21 mostra um fluxograma 2100 de processamento de migração de metadados que pode ser realizada pela lógica de migração de metadados 1224. A lógica de migração de metadados 1224 pode executar, por exemplo, o processamento durante um processo de mapeamento, tais com os processos de mapeamento 800 e 1800. Este processamento de migração de metadados pode ajudar no movimento de todos os metadados existentes ou quaisquer metadados despercebidos de um GAP para um outro ou de uma versão de GAP para uma outra.

[00175] A lógica de migração de metadados 1224 primeiramente pode determinar se o elemento de GUI de origem tem metadados associados a ele (2102). Por exemplo, a lógica de migração de metadados 1224 pode usar um alias de GAP de origem e um identificador de elemento de GUI de origem para a localização de um registro de metadados de elemento de GUI de origem. Então, a lógica de migração de metadados 1224 pode buscar dentro daquele registro de metadados de elemento de GUI de origem quanto a quaisquer campos de metadados relevantes, tal como um campo de tipo 1326, um campo de notação 1328 ou campo de outros metadados 1332. Se a lógica de migração de metadados 1224 determinar que o elemento de GUI de origem não tem metadados relevantes associados a ele, a lógica de migração de metadados 1224 então poderá determinar se o elemento de GUI de destino tem metadados associados a ele (2104). Esta determinação pode ser realizada de uma maneira similar a se determinar se o elemento de GUI de origem tinha metadados associados a ele (2102). Se a lógica de migração de metadados 1224

determinar que o elemento de GUI de destino não tem metadados relevantes associados a ele, a lógica poderá terminar.

[00176] Se a lógica de migração de metadados 1224 determinar que o elemento de GUI de origem tem metadados associados a ele, a lógica de migração de metadados 1224 determinará se um elemento de GUI de destino tem metadados associados a ele (2106). Esta determinação pode ser realizada de uma maneira similar a determinar se o elemento de GUI de origem tinha metadados associados a ele (2102). Se a lógica de migração de metadados 1224 determinar que o elemento de GUI de destino não tem metadados associados a ele, a lógica de migração de metadados 1224 poderá prover um alerta para o processamento adicional de instruções (2108). O alerta pode pedir instruções quanto a se é para copiar metadados a partir do elemento de GUI com metadados para o registro de elemento de GUI sem os metadados (2110). Se a resposta ao alerta for 'não', então a lógica poderá terminar. Se a resposta ao alerta for 'sim', então a lógica poderá realizar o processo de cópia (2112), antes de terminar. Um processo similar ocorre quando um elemento de GUI de origem não tem metadados, mas um elemento de GUI de destino tem.

[00177] Se a lógica de migração de metadados 1224 determinar que um elemento de GUI de origem e um elemento de GUI de destino têm, cada um, metadados associados a eles, a lógica de migração de metadados 1224 poderá prover um alerta para o processamento adicional de instruções (2114). Por exemplo, o alerta pode incluir opções para Sobrescrever um dos conjuntos de metadados com um outro, Apensar um conjunto de metadados com o outro (ou, opcionalmente, apensar cada conjunto de metadados ao outro), ou Mover um conjunto de metadados para um registro de metadados de elemento de GUI completamente diferente (2116). Se a resposta ao alerta incluir Sobrescrever um dos conjuntos de metadados com o

outro, então, a lógica de migração de metadados 1224 poderá realizar o processo de sobrescrita (2118), antes de terminar. Se a resposta ao alerta incluir Apensar um conjunto de metadados ao outro, então, a lógica de migração de metadados 1224 poderá realizar o processo de apensar (2120), antes de terminar. Se a resposta ao alerta incluir Mover um conjunto de metadados ao outro, então, a lógica de migração de metadados 1224 poderá realizar o processo de mover (2120), antes de terminar. De forma alternativa ou adicional, a lógica de migração de metadados 1224 pode realizar o processo de mover (2122) e, então, prover um outro alerta para uma ação continuada, tal como copiar um conjunto de metadados no registro de metadados deixado vago pelo processo de mover, de modo similar ao alerta de copiar (2110).

[00178] A Figura 22 mostra um fluxograma 2200 de processamento de migração de metadados que pode ser realizado pela lógica de migração de metadados 1224. A lógica de migração de metadados 1224 pode primeiramente buscar elementos de GUI que tenham metadados associados a eles (2202). Por exemplo, esta busca pode ser realizada pelo acesso a um registro de metadados de GAP ou de versão de GAP e acessando-se cada registro de metadados de elemento de GUI armazenado naquele registro. De forma alternativa ou adicional, a lógica de migração de metadados 1224 pode consultar uma lista de GAPs ou de versões de GAP juntamente com os identificadores de elemento de GUI associados e acessar os registros de metadados de elemento de GUI para cada um dos identificadores de elemento de GUI individualmente. De forma alternativa ou adicional, a lógica de migração de metadados 1224 pode realizar uma técnica de busca primeiro em profundidade, busca primeiro em largura ou outra técnica de busca para se acessarem todos os registros de metadados de elemento de GUI em um depósito de metadados. Uma

vez que a lógica de migração de metadados 1224 tenha adquirido um elemento de GUI, ela pode determinar se o elemento de GUI tem metadados apropriados de uma maneira similar a determinar se o elemento de GUI de origem tem metadados associados a ele (2102).

[00179] Uma vez que a lógica de migração de metadados 1224 tenha identificado um elemento de GUI com metadados associados a ele, a lógica de metadados 1224 pode determinar se o elemento de GUI tem um mapeamento associado a ele (2204). Por exemplo, esta determinação pode incluir olhar para o campo de mapeamento de elemento de GUI 1330 no registro de metadados de elemento de GUI. De forma alternativa ou adicional, a determinação pode ter sido feita por um outro processo lógico com o resultado a partir da lógica de migração de metadados 1224 passado juntamente com a identificação do elemento de GUI.

[00180] Se a lógica de migração de metadados 1224 determinar que o elemento de GUI tem um mapeamento associado, a lógica de migração de metadados 1224 poderá assumir que um outro processo de lógica de migração de metadados, tal como o processo de lógica 2100, já migrou quaisquer metadados relevantes e, assim, em seguida determinará se o elemento de GUI foi o último elemento que precisou ser checado quanto a uma migração de metadados (2206). Esta determinação pode incluir olhar para o próximo GAP ou versão de GAP e o identificador de elemento de GUI associado em uma lista. De forma alternativa ou adicional, a determinação pode incluir olhar para o próximo registro de metadados de elemento de GUI criado em um algoritmo de busca primeiro em profundidade, primeiro em largura ou outro apropriado. Se a lógica de migração de metadados 1224 determinar que o elemento de GUI foi o último elemento de GUI a processar, então, a lógica de migração de metadados 1224 poderá terminar.

[00181] Se a lógica de migração de metadados 1224 determinar que o elemento de GUI não foi o último elemento de GUI a ser processado, a lógica de migração de metadados 1224 poderá se mover para o próximo elemento de GUI (2208). Este movimento pode incluir acessar o próximo GAP ou versão de GAP e o identificador de elemento de GUI associado em uma lista. De forma alternativa ou adicional, o movimento pode incluir acessar o próximo registro de metadados de elemento de GUI criado em um algoritmo de busca primeiro em profundidade, primeiro em largura ou outro apropriado. Após o movimento, a lógica de migração de metadados 1224 pode circular de volta e determinar se o novo elemento de GUI tem um mapeamento associado a ele (2204).

[00182] Se a lógica de migração de metadados 1224 determinar que um elemento de GUI não tem um mapeamento associado a ele, então, a lógica de migração de metadados 1224 poderá prover um alerta por instruções adicionais (2210). O alerta pode requisitar instruções adicionais quanto a se é para mapear o elemento de GUI para um outro elemento de GUI (2212). Se a resposta ao alerta incluir um 'sim', então, a lógica de migração de metadados 1224 poderá ativar uma lógica de processamento de mapeamento 1220 e terminar (2214). De forma alternativa ou adicional, a lógica de migração de metadados 1224 pode ativar uma lógica de processamento de mapeamento 1220 e um processamento de migração de metadados 2100, antes de terminar.

[00183] Se a resposta ao alerta 1312 incluir um 'não', então, a lógica de migração de metadados 1224 poderá determinar se o elemento de GUI foi o último elemento que precisava ser checado quanto a uma migração de metadados (2216), de modo similar à checagem 2206. SE a lógica de migração de metadados 1224 determinar que o elemento de GUI foi o último elemento de GUI a

processar, então, a lógica de migração de metadados 1224 poderá terminar. Se a lógica de migração de metadados 1224 determinar que o elemento de GUI não foi o último elemento de GUI a ser processado, a lógica de migração de metadados 1224 poderá se mover para o próximo elemento de GUI (2208).

[00184] A Figura 1 mostra um sistema comparador de aplicativo de interface gráfica de usuário (GUI) ("sistema") 106. O sistema 106 pode incluir um construtor de modelo de GUI 154 e um comparador de GUI 160. O construtor de modelo de GUI 154 pode aceitar uma versão 'n' de aplicativo de GUI (GAP) ("Vn de GAP") 150 e criar um modelo de GUI de Vn de GAP 156. O construtor de modelo de GUI 154 também pode aceitar uma versão de GAP 'n+1' ("Vn+1 de GAP") 152 e criar um modelo de GUI de Vn+1 de GAP 158. O comparador de GUI 160 pode aceitar o modelo de GUI de Vn de GAP 156 e o modelo de GUI de Vn+1 de GAP 158, bem como uma informação recebida através da lógica de comunicação 163 para a criação de um modelo de diferença de GUI 162. O modelo de diferença de GUI 162 pode ser enviado para outros sistemas através da lógica de comunicação 163. Os modelos de GUI de GAP 156 e 158 podem ter uma estrutura plana, de árvore, hierárquica, aninhada ou outro tipo de estrutura. A Vn de GAP 150 pode ser uma versão atual de um GAP em particular, enquanto a Vn+1 de GAP 152 pode ser uma versão subsequente de GAP. O modelo de GUI de Vn de GAP 156 pode prover uma representação da estrutura de GUI da versão de GAP atual, enquanto o modelo de GUI de Vn+1 de GAP 158 pode prover uma representação da estrutura de GUI da versão de GAP subsequente.

[00185] A Figura 23 mostra uma implementação do sistema comparador de aplicativo de GUI 106. O sistema 106 pode incluir um processador 2302, uma memória 2304 e um visor 2306. O sistema 106 pode trocar uma informação com outros sistemas, tal como um

depósito de dados de elemento de GUI ("depósito") 138, um analisador de script 170 e uma rede 2314 através de uma lógica de comunicação 163. A lógica de comunicação 163 pode ser uma interface com fio/ sem fio, um mecanismo de comunicação interprocesso, uma memória compartilhada, uma interface de serviços da web, ou quaisquer outros tipos de interface de comunicação. O depósito 138 pode incluir os mapeamentos de elemento de GUI de GAP 1308. A rede 2314 pode se conectar a terminais locais ou remotos 2318, para uma interação de operador local ou remoto com o sistema 106, a outros processos rondando em outras máquinas ou a outras entidades que interajam com o sistema 106.

[00186] A memória 2304 pode incluir uma lógica de construtor de modelo de GUI 2320. A lógica de construtor de modelo de GUI 2320 pode se comunicar com um proxy 2322. O proxy 2322 pode ser armazenado na memória 2304 e acessar uma tabela de GAP 2324. O proxy 2322 pode se comunicar com GAPs, tal como uma versão de GAP atual 2326 e uma versão de GAP subsequente 2328. A versão de GAP atual 2326 e a versão de GAP subsequente 2328 já podem residir na memória 2304. De forma alternativa ou adicional, o sistema 106 pode requisitar e receber a versão de GAP atual 2326 e a versão de GAP subsequente 2328 através da lógica de comunicação 163, mediante o que a versão de GAP atual 2326 e a versão de GAP subsequente 2328 podem ser armazenadas na memória 2304.

[00187] O proxy 2322 pode incluir uma lógica que insere os ganchos 2330 e 2332 em um espaço de processo dos GAPs 2326 e 2328. O proxy 2322 pode se comunicar com os ganchos 2330 e 2332. Em particular, o proxy 2322 pode trocar mensagens com os ganchos 2330 e 2332 para a obtenção do estado de qualquer um ou de todos os elementos de GUI nos GAPs 2326 e 2328. Os ganchos 2330 e 2332 podem ser programas que respondem a mensagens a partir do

proxy 2322 e podem interagir através de uma camada de acessibilidade 2334 de um sistema operacional 2336 para a descoberta e o relatório de uma informação sobre os elementos de GUI nos GAPs 2326 e 2328 para o proxy. A camada de acessibilidade 2334 pode expor uma interface de acessibilidade através da qual o proxy 2322 e os ganchos 2330 e 2332 podem invocar métodos e regular e recuperar valores e características de elementos de GUI e, desse modo, selecionar, destacar, controlar, modificar, atribuir identificadores para ou interagir de outra forma com os elementos de GUI nos GAPs.

[00188] A camada Microsoft (TM) Active Accessibility (MSAA) é um exemplo de uma camada de acessibilidade adequada. Nesse sentido, os GAPs 2326 e 2328 expõem as interfaces de acessibilidade que exportam métodos para acesso e manipulação das propriedades e do comportamento de elementos de GUI. Por exemplo, os GAPs 2326 e 2328 podem empregar a interface IAccessible para se permitirem acesso e controle em relação ao elemento de GUI usando chamadas de interface de programação de aplicativo (API) de MSAA. A interface IAccessible ainda facilita aplicativos para exposição de uma árvore de nós de dados que constituem cada janela na interface de usuário com a qual se interage atualmente. A lógica de construtor de modelo de GUI 2320 e o proxy 2322 podem incluir, então, declarações de programa para acesso e controle do elemento de GUI, como se o elemento de GUI fosse um objeto de programação convencional. As chamadas de API de acessibilidade podem incluir: realizar ações em objetos, obter valores de objetos, regular valores em objetos, navegar para objetos, e regular propriedades em objetos, e outras chamadas.

[00189] O proxy 2322 pode ser um programa daemon e pode começar antes da lógica de construtor de modelo de GUI 2320. O proxy 2322 pode estar ciente de um ou mais GAPs. Quando o proxy

2322 começa, ele carrega a tabela de GAP 1106, a qual pode incluir um conjunto predefinido de entradas de GAP quanto às quais o proxy 2322 está ciente. Uma entrada de GAP pode assumir a forma:

[00190] <Alias, <File0, Path0, Dir0, CommandLine0>, <File1, Path1, Dir1, CommandLine1>>

[00191] onde Alias é um nome predefinido único para o GAP (por exemplo, um nome genérico para a versão de GAP atual 2326 e a versão de GAP subsequente 2328), File0 é o nome do programa executável para a versão de GAP atual 2326, Path0 é o caminho absoluto para File0, Dir0 é o caminho absoluto para o diretório a partir do qual File0 deve ser executado, e CommandLine especifica os argumentos de linha de comando para File0. File1, Path1, Dir1 e CommandLine1 provêm parâmetros similares para a versão de GAP subsequente 2328.

[00192] Quando a lógica de construtor de modelo de GUI 2320 começa, ela pode se conectar ao proxy 2322. Uma vez conectada, a lógica de construtor de modelo de GUI 2320 requisita a tabela de GAP 2324 ao enviar uma mensagem de requisição de tabela de GAP para o proxy 2322. O proxy 2322 pode responder pelo envio de uma mensagem de resposta de tabela de GAP incluindo a tabela de GAP 2324 para a lógica de construtor de modelo de GUI 2320. Uma troca de mensagem de exemplo é mostrada na Tabela 4:

Tabela 4
mensagem de requisição de tabela de GAP <GetGapTable/>
mensagem de resposta de tabela de GAP
<GapTable> <GAP Alias = "name" <V_N File="gap.exe" Path="c:\path\N" CommandLine="-c1"/> <V_N1 File="gap.exe" Path="c:\path\N1" CommandLine="-c1"/> </GAP> </GapTable>

[00193] A lógica de construtor de modelo de GUI 2320 então pode

prover uma lista de GAPs a partir da qual um operador pode escolher. O operador pode acessar o sistema 106 localmente através do visor 2306 ou remotamente, por exemplo, através do terminal 2318. A lógica de construtor de modelo de GUI 2320 então pode criar uma mensagem de carregar GAP, por exemplo, <LoadGap Alias="name"/> e enviar a mensagem de carregar GAP para o proxy 2322 para se começar qualquer GAP selecionado (o qual então pode exibir sua interface de usuário). Uma mensagem de carregar GAP pode fazer com que o proxy 2322 comece múltiplas versões de um GAP identificado em conjunto com a tabela de GAP 2324 na seção de <GAP>.

[00194] Após começar os GAPs, o proxy 2322 pode injetar ganchos no espaço de processo de GAPs. O gancho pode se conectar ao proxy 2322 e enviar uma mensagem de confirmação (por exemplo, <GAP File="gap.exe" Instance="192"/>). O proxy 2322 pode enviar uma mensagem de sucesso (por exemplo, <Loaded Alias="name" VN="192" VN1="193"/>) para a lógica de construtor de modelo de GUI 2320, desse modo reconhecendo que os GAPs foram começados de forma bem-sucedida.

[00195] A lógica de construtor de modelo de GUI 2320 pode requisitar o estado atual de cada GAP começado. Nesse sentido, a lógica de construtor de modelo de GUI 2320 pode enviar uma mensagem de requisição de estado (por exemplo, <GetState Alias="name"/>) para o proxy 2322. Por sua vez, o proxy 2322 pode localizar a conexão com os ganchos correspondentes dos GAPs e enviar uma mensagem de requisição de estado (por exemplo, <GetState/>) para os ganchos. Os ganchos podem criar um estado de GAP (incluindo identificadores únicos para os elementos de GUI), tal como uma árvore de estado, codificá-lo (por exemplo, em um formato em XML), e o enviar para o proxy 2322. O proxy 2322 pode

encaminhar o estado de GAP para a lógica de construtor de modelo de GUI 2320. Uma mensagem de estado de GAP de exemplo enviada pelo proxy 2322 é mostrada na Tabela 5.

Tabela 5

mensagem de estado de GAP

```
<State SeqNumber="1" Name="name" Alias="name" ProcessID="972">
        <GUIElement Alias="name">
                <Location x="15" y="200" width="23" height="98"/>
                <Description>Action</ Description>
                <DefAction>Action</DefAction>
                <UniqueID>0xcafebabe</UniqueID>
                <Class>LISTBOX</Class>
                <Values>
                        <Value SeqNumber="1">someval</Value>
                        ……
                </Values>
        </GUIElement>
        ……
</State>
```

[00196] O estado de GAP contém uma informação sobre os elementos de GUI compondo uma dada tela, bem como os valores destes elementos e seus identificadores atribuídos. O estado de GAP especifica os elementos de GUI de GAP e os valores dos elementos de GUI. Em uma implementação, o estado de GAP é refletido em uma estrutura de linguagem de marcação extensível (XML), onde o elemento 'State' tem um ou mais elementos filhos 'GAP', cujos elementos filhos por sua vez são 'GUIElements'. Por exemplo, os elementos de GUI podem ser containeres ou básicos. Os elementos de GUI de container contêm outros elementos, enquanto os elementos básicos não contêm outros elementos. A estrutura de XML reflete a hierarquia de contenção ao permitir que os GUIElements contenham outros GUIElements.

[00197] Na estrutura de XML, o atributo SeqNumber pode designar um número de sequência único do estado dentro do GAP. Uma vez que os estados são mapeados para telas de GUI, a cada estado pode

ser dado um nome o qual é especificado pelo atributo 'Name'. Os atributos Alias e ProcessID podem denotar o alias do GAP e seu identificador de processo de instância, respectivamente. O identificador de processo de instância pode diferenciar entre a versão de GAP atual e a versão de GAP subsequente.

[00198] A lógica de construtor de modelo de GUI 2320 pode construir modelos de GUI de GAP com base em mensagens de estado de GAP recebidas a partir do proxy 2322. Por exemplo, a lógica de construtor de modelo de GUI 2320 pode construir um modelo de GUI de Vn de GAP 2338 a partir de mensagens de estado de GAP referentes à versão de GAP atual 2326. De modo similar, a lógica de construtor de modelo de GUI 2320 pode construir um modelo de GUI de Vn+1 de GAP 2340 a partir de mensagens de estado de GAP referentes à versão de GAP subsequente 2328.

[00199] O processador 2302 pode invocar a lógica de comparação de GAP 2342 armazenada na memória 2304. A lógica de comparação de GAP 2342 pode comparar dois modelos de GUI de GAP, tais como os modelos de GUI de GAP 2338 e 2340 e produzir um modelo de diferença de GUI 2344. A lógica de comparação de GAP 2342 pode incluir uma lógica de recuperação de mapeamento 2346, uma lógica de travessia de representação 2348, uma lógica de análise ponderada 2350 e uma lógica de construção de combinação 2352.

[00200] A lógica de recuperação de mapeamento 2346 pode requisitar mapeamentos de elemento de GUI de GAP em particular a partir dos mapeamentos de elemento de GUI de GAP 1308 em um depósito de dados de elemento de GUI 138 e armazenar os mapeamentos de elemento de GUI de GAP em particular na memória 2304 como mapeamentos de versão de elemento de GUI 2354.

[00201] A lógica de travessia de representação 2348 pode atravessar um modelo de GUI de GAP, tal como o modelo de GUI de

Vn de GAP 2338. Por exemplo, a lógica de travessia de representação 2348 pode determinar o próximo nó a visitar nos modelos de GUI de GAP 2338 e 2340. De forma alternativa ou adicional, a lógica de travessia de representação 2348 pode atravessar todo um ou partes de um modelo de diferença de GUI, tal como o modelo de diferença de GUI 2344. O próximo nó a visitar pode ser determinado, como exemplos, de forma de primeiro profundidade ou primeiro largura.

[00202] A lógica de análise ponderada 2350 pode usar os pesos característicos de GUI 2356 obtidos a partir de uma tabela de peso 2358 para a determinação de um valor de similaridade entre um elemento de GUI em um primeiro modelo de GUI de GAP, tal como o modelo de GUI de Vn de GAP 2338, e cada elemento de GUI em um segundo modelo de GUI de GAP, tal como o modelo de GUI de Vn+1 de GAP 2340. Pesos característicos de GUI diferentes 2356 podem ser atribuídos às similaridades ou diferenças entre características de elemento de GUI diferentes 2360 ou propriedades, que podem estar presentes ou ausentes entre os elementos de GUI nos dois modelos de GUI de GAP. As características de elemento de GUI 2360 podem incluir características de elemento de GUI tais como tamanho, posição XY, cor, posição de janela, tipo de fonte, tamanho de fonte, estilo de borda, cor de fundo, cor de primeiro plano, apenas de leitura/apenas de escrita ou qualquer outra característica de elemento de GUI. De forma alternativa ou adicional, as características de elemento de GUI 2360 podem incluir um Papel de camada de acessibilidade, Classe, Estilo, Estilo Estendido, número de filhos, nível em uma árvore ou hierarquia, Nome do elemento de GUI, ou outras propriedades atribuídas a uma camada de acessibilidade. A tabela de peso também pode incluir notas 2362 associadas aos pesos 2356 atribuídos às características de GUI 2360 que podem explicar o raciocínio por trás de cada valor de peso.

[00203] A lógica de análise ponderada 2350 pode armazenar um escore 2364 em uma tabela de escore 2366 com base em cada valor de similaridade gerado pela lógica de análise ponderada 2350. Cada escore 2364 na tabela de escore 2366 pode corresponder a um identificador de origem 2368 e um identificador de destino 2370. O identificador de origem 2368 e o identificador de destino 2370 podem ser um valor único ou uma combinação de valores (por exemplo, incluindo aliases de GAP) identificando os GAPs e os elementos de GUI que a lógica de análise ponderada 2350 comparou para o cálculo de cada escore 2364.

[00204] A lógica de construção de combinação 2352 pode comparar os valores de similaridade gerados pela lógica de análise ponderada 2350 e/ou os escores 2364 armazenados na tabela de escore 2366 em relação a um limite de similaridade 2372. Esta comparação pode determinar se dois elementos de GUI são suficientemente similares para serem considerados uma combinação de uma versão de GAP atual para a versão de GAP subsequente. A lógica de construção de combinação 2352 pode criar um link entre os elementos de GUI combinando no modelo de diferença de GUI 2344. O link pode ser armazenado na representação de elemento de GUI dentro do modelo de diferença de GUI 2344 como um link de elemento de GUI 2374 com um escore de combinação correspondente opcional 2376. O link de elemento de GUI pode compreender um identificador de um segundo elemento de GUI 2377. O identificador pode ser o identificador de origem 2368, o identificador de destino 2370, ou ambos.

[00205] Em operação, a lógica de comparação de GAP 2342 pode obter os modelos de GUI de GAP 2338 e 2340 pela recuperação deles a partir da memória 2304, ao chamar a lógica de construtor de modelo de GUI 2320 ou de uma outra maneira. A lógica de comparação de GAP 2342 pode criar um modelo de diferença de GUI de base como

um nó de raiz a partir do qual os modelos de GUI de GAP 2338 e 2340 descem em ramificações diferentes a partir da raiz. A lógica de comparação de GAP 2342 então pode determinar o próximo nó a visitar em cada um dos modelos de GUI de GAP usando-se a lógica de travessia de representação 2348.

[00206] A lógica de comparação de GAP 2342 pode iniciar a execução da lógica de recuperação de mapeamento 2346 para a obtenção dos mapeamentos de versão de elemento de GUI disponíveis a partir de fontes externas, tal como o depósito de metadados 138. A lógica de comparação de GAP 2342 pode requisitar todos os mapeamentos de versão de elemento de GUI disponíveis, ou pode requisitar especificamente mapeamentos de versão de elemento de GUI para o próximo nó no modelo de GUI de GAP atual. Se um mapeamento de versão de elemento de GUI estiver disponível para o próximo nó no modelo de GUI de GAP atual, a lógica de comparação de GAP 2342 pode abrir mão da execução da lógica de análise ponderada 2350. Ao invés disso, a lógica de comparação de GAP 2342 pode empregar a lógica de construção de combinação para escrever um link de elemento de GUI para o modelo de diferença de GUI de base. Como uma outra alternativa, quando um mapeamento de versão de elemento de GUI está disponível, a lógica de comparação de GAP 2342 pode criar uma entrada correspondente na tabela de escore 2366, com base na informação disponível no mapeamento de versão de elemento de GUI.

[00207] Contudo, a lógica de comparação de GAP 2342 não precisa abrir mão da análise ponderada, quando existir um mapeamento de versão de elemento de GUI. Ao invés disso, a lógica de comparação de GAP 2342 pode decidir quando prosseguir com o mapeamento ponderado com base no nível de confiança provido no mapeamento de versão de elemento de GUI. Por exemplo, quando o nível de confiança

excede a um limite de confiança, a lógica de comparação de GAP 2342 pode abrir mão da execução da análise ponderada. Como um outro exemplo, quando o nível de confiança especifica um mapeamento manual, a lógica de comparação de GAP 2342 pode abrir mão da execução da análise ponderada.

[00208] A lógica de comparação de GAP 2342 usa a lógica de análise ponderada 2350 para determinar valores de similaridade entre cada elemento de GUI no modelo de GUI de GAP atual 2338 e cada elemento no modelo de GUI de GAP subsequente 2340. A lógica de análise ponderada 2350 é descrita em maiores detalhes abaixo. A lógica de análise ponderada registra os valores de similaridade na tabela de peso 2358 como escores. Os escores podem ser os valores de similaridade, valores de similaridade normalizados, ou baseadas de alguma outra forma nos valores de similaridade.

[00209] Tendo determinado os valores de similaridade, a lógica de comparação de GAP 2342 pode usar a lógica de construção de combinação 2352 para determinar se elementos de GUI combinam entre as versões de GAP. Para essa finalidade, a lógica de construção de combinação 2352 pode comparar os escores na tabela de escore em relação ao limite de similaridade 2372. Os elementos de GUI com escores que excedam ao limite de similaridade 2372 podem ser considerados combinações sob a hipótese de que quanto mais alto o escore de similaridade, mais provavelmente ele se referirá aos elementos de GUI correspondentes. A lógica de construção de combinação 2352 pode criar links de elemento de GUI no modelo de diferença de GUI de base quando combinações forem determinadas.

[00210] A Figura 24 mostra um exemplo de uma porção 2400 de um modelo de GUI de GAP atual. A porção 2400 usa uma representação em XML, mas qualquer outra representação pode ser usada, ao invés disso. A porção 2400 inclui um alias de GAP 2402

("University Directory0"), um identificador de elemento de GUI 2404 ("0x90b52"), e características de GUI 2406. As características de GUI 2406 incluem uma localização com um valor x de "173", um valor y de "486", um valor de largura de "336" e um valor de comprimento de "274". Outras características de GUI 2406 incluem um valor de classe de "WindowsForms10.LISTBOX.app4", um valor de estilo de "0x560100c1" e um valor de estilo estendido de "0xc0000a00".

[00211] A Figura 25 mostra um exemplo de uma porção correspondente 2500 de um modelo de GUI de GAP subsequente usando uma representação em XML. A porção correspondente 2500 inclui um alias de GAP 2302 ("University Directory1"), um identificador de elemento de GUI 2504 ("0x90e66"), e características de GUI 2506. As características de GUI 2406 incluem uma localização com um valor x de "248", um valor y de "653", um valor de largura de "299" e um valor de comprimento de "24". Outras características de GUI 2506 incluem um valor de classe de "WindowsForms10.COMBOBOX.app.0.378734a", um valor de estilo de "0x560100242" e um valor de estilo estendido de "0xc0000800".

[00212] O elemento de GUI com o identificador 2504 é uma versão modificada do elemento de GUI com o identificador 2404. Em outras palavras, quando o programador projetou a versão de GAP subsequente, o programador modificou o elemento de GUI com identificador 2404 para obter o elemento de GUI com identificador 2504. Em particular, as Figuras 24 e 25 mostram que as mudanças a seguir foram feitas: o estilo mudou de "0x560100c1" para "0x560100242" e o estilo estendido mudou de "0xc0000a00" para "0xc0000800". Estas diferenças nas características de elemento de GUI não são prontamente discerníveis para programadores de script de teste. Contudo, outras mudanças podem ser discerníveis por um programador, tal como a mudança de classe de

[00213] "WindowsForms10.LISTBOX.app4" para

[00214] "WindowsForms10.COMBOBOX.app.0.378734a".

[00215] A Figura 26 mostra um exemplo de uma porção de diferença 2600 de um modelo de diferença de GUI. A porção de diferença 2600 pode incluir uma seção de versão atual 2602, incluindo uma informação de elemento de GUI retirada do modelo de GUI de GAP atual (por exemplo, partes de porção 2400) e uma seção de versão subsequente correspondente 2604, incluindo uma informação de elemento de GUI retirada do modelo de GUI de GAP subsequente (por exemplo, partes de porção correspondente 2500). As seções 2602 e 2064 podem ser separadas por um primeiro elemento de versão 2606 e um segundo elemento de versão 2608. Neste exemplo, o primeiro elemento de versão 2606 tem um valor de "0", enquanto o segundo elemento de versão 2608 tem um valor de "1". O valor de elemento de versão "0" pode indicar que a seção se originou de um modelo de GUI de GAP atual, onde o valor de elemento de versão "1" pode indicar que a seção se originou de um modelo de GUI de GAP subsequente.

[00216] O modelo de diferença de GUI 2344 pode ser em uma configuração plana, onde o modelo de diferença de GUI 2344 inclui uma seção única 2602 e uma seção correspondente única 2604. De forma alternativa ou adicional, o modelo de diferença de GUI 2344 pode estar em uma configuração de árvore, hierárquica ou aninhada, onde o modelo de diferença de GUI 2344 inclui múltiplas seções 2602 e múltiplas seções correspondentes 2604. Na configuração de árvore, hierárquica ou aninhada, elementos de GUI similares podem ser representados por um nó único. De forma alternativa ou adicional, elementos de GUI similares podem ser representados em nós separados. O modelo de diferença de GUI 2344 pode incluir todos os elementos de GUI de ambos os modelos de GUI de GAP.

Alternativamente, o modelo de diferença de GUI 2344 pode incluir apenas os elementos do segundo modelo de GUI de GAP e as porções correspondentes do primeiro modelo de GUI de GAP. O modelo de diferença de GUI 2344 pode ser formado com base em um algoritmo de bissimulação, conforme descrito em maiores detalhes abaixo.

[00217] A lógica de comparação de GAP 2342 pode criar a porção de diferença 2600 no formato apresentado na Figura 26, quando combinar o modelo de GUI de GAP de origem com o modelo de GUI de GAP de destino sob um nó de raiz. De forma alternativa ou adicional, a lógica de comparação de GAP 2342 pode criar a porção de diferença 2600 após a lógica de recuperação de mapeamento 2346 obter um mapeamento relevante entre o elemento de GUI representado na seção de versão atual 2602 e a seção de versão subsequente 2604. De forma alternativa ou adicional, a lógica de comparação de GAP 2342 pode criar a porção de diferença 2600 após a lógica de construção de combinação 2352 criar um link entre a seção de versão atual 2602 e a seção de versão subsequente 2604.

[00218] A Figura 27 mostra um fluxograma 2700 para a lógica de construtor de modelo de GUI 2320. A lógica de construtor de modelo de GUI 2320 pode exibir uma lista de GAPs para escolha pelo operador. Conforme citado acima, a lista de GAPs pode ser estabelecida na tabela de GAP 2324. A lógica de construtor de modelo de GUI 2320 monitora quanto a uma entrada de operador (2702) para a seleção de um GAP. A lógica de construtor de modelo de GUI 2320 desse modo determina uma versão de GAP selecionada (2704).

[00219] A lógica de construtor de modelo de GUI 2320 então pode criar uma mensagem de carregamento de GAP, por exemplo, <LoadGap Alias="name"/> e enviar a mensagem de carregamento de GAP para o proxy 2322 para começar a versão de GAP selecionada, a

qual então pode exibir seu GUI (2706). Após o começo do GAP, o proxy 2322 pode injetar um gancho no espaço de processo de GAP (2708). O gancho pode se conectar ao proxy 2322 e enviar uma mensagem de confirmação (por exemplo, <GAP File="gap.exe" Instance="192"/>). O proxy 2322 pode enviar uma mensagem de sucesso (por exemplo, <Loaded Alias="name" VN="192" VN1="193"/>) para a lógica de construtor de modelo de GUI 2320, desse modo reconhecendo que o GAP foi começado de forma bem-sucedida.

[00220] A camada de acessibilidade, o proxy, o gancho e a lógica de construtor de modelo de GUI 2320 monitoram uma interação de operador com elementos de GUI na versão de GAP selecionada (2710). A lógica de construtor de modelo de GUI 2320 pode enviar uma mensagem de requisição de estado <GetState Alias="name"/>) para o proxy 2322 para obtenção da informação de elemento de GUI a partir do gancho (2712). Por sua vez, o proxy 2322 pode localizar a conexão com o gancho correspondente na versão de GAP selecionada e enviar uma mensagem de requisição de estado (por exemplo, <GetState/>) para o gancho. O ganho pode criar um estado de GAP (incluindo identificadores únicos para elementos de GUI), tal como uma árvore de estado, codificá-lo (Por exemplo, em um formato em XML), e enviá-lo para o proxy 2322. O proxy 2322 pode encaminhar o estado de GAP para a lógica de construtor de modelo de GUI 2320. A informação de elemento de GUI pode ser retornada para a lógica de construtor de modelo de GUI 2320 uma tela de uma vez, um elemento de GUI de uma vez, um aplicativo inteiro de uma vez, ou em alguma outra segmentação discreta do GAP.

[00221] A finalidade de monitoração da interação de operação com o GAP é permitir que a lógica de construtor de modelo de GUI 2320 registre as estruturas das telas e as ações de operador nos GAPs. A lógica de construtor de modelo de GUI 2320 intercepta eventos de

operador usando a camada de acessibilidade. Através destes eventos, a lógica de construtor de modelo de GUI 2320 registra a sequência de telas através do que o operador navega, bem como as ações que o operador realiza em elementos de GUI. Quando registrando a sequência de telas, a lógica de construtor de modelo de GUI 2320 obtém uma informação sobre a estrutura do GAP e as propriedades dos elementos de GUI individuais usando as interfaces de acessibilidade habilitada. Assim sendo, a lógica de construtor de modelo de GUI 2320 extrai dados estruturais de elemento de GUI (2714) e características de elemento de GUI (2716) a partir da informação retornada pela camada de acessibilidade. A lógica de construtor de modelo de GUI 2320 usa os dados estruturais de elemento de GUI e as características de elemento de GUI para adicionar uma informação de elemento de GUI em um modelo de GUI de GAP (2718), por exemplo, em uma base de elemento por elemento, tela por tela ou em outra base em incrementos. A lógica de construtor de modelo de GUI 2320 pode continuar a construir o modelo de GUI de GAP até o operador parar de interagir com o GAP selecionado (2720).

[00222] O modelo de GUI de GAP que resulta pode ser uma captura plena ou parcial da estrutura de GUI de GAP inteira. Assim, quando o operador está interessado em comparar pedaços específicos de uma GUI entre dois GAPs, o operador pode exercitar apenas aqueles pedaços de interesse. A lógica de construtor de modelo de GUI 2320 captura os pedaços específicos em um modelo de GUI de GAP específico para os pedaços que o operador exercitou, ao invés de todo aspecto de todo elemento de GUI no GAP selecionado inteiro. O operador pode rodar a versão de GAP atual 2326 e a versão de GAP subsequente 2328 com a lógica de construtor de modelo de GUI 2320 para criar o modelo de GUI de Vn de GAP 2338 e o modelo de

GUI de Vn+1 de GAP 2340, respectivamente.

[00223] A Figura 28 mostra um fluxograma 2800 para uma lógica de comparação de GAP, tal como a lógica de comparação de GAP 2342. A lógica de comparação de GAP 2342 pode receber um primeiro modelo de GUI de GAP (2802). Por exemplo, a lógica de comparação de GAP 2342 pode acessar o modelo de GUI de Vn de GAP 2338 armazenado na memória 2304. A lógica de comparação de GAP 2342 pode receber um segundo modelo de GUI de GAP (2804). Por exemplo, a lógica de comparação de GAP 2342 pode acessar o modelo de GUI de Vn+1 de GAP 2340 armazenado na memória 2304. De forma alternativa ou adicional, a lógica de comparação de GAP 2342 pode requisitar e receber um primeiro e um segundo modelos de GUI de GAP a partir da lógica de comunicação 163.

[00224] A lógica de comparação de GAP 2342 então pode combinar o primeiro modelo de GUI de GAP e o segundo modelo de GUI de GAP para a criação do modelo de diferença de GUI (2806). Os primeiro e segundo modelos de GUI de GAP podem ser combinados em uma configuração plana. De forma alternativa ou adicional, os primeiro e segundo modelos de GUI de GAP podem ser combinados em uma configuração de árvore, hierárquica ou aninhada.

[00225] A lógica de comparação de GAP 2342 pode invocar, então, a lógica de recuperação de mapeamento 2346. A lógica de recuperação de mapeamento 2346 pode determinar se os mapeamentos de versão de elemento de GUI estão disponíveis para o primeiro modelo de GUI de GAP e o segundo modelo de GUI de GAP (2808). Esta determinação pode ser realizada ao se consultar uma fonte local, tal como a memória 2304, quanto a mapeamentos de versão de elemento de GUI. De forma alternativa ou adicional, a lógica de recuperação de mapeamento 2346 pode consultar um depósito de dados de elemento de GUI 138 através da lógica de comunicação 163.

[00226] Se a lógica de recuperação de mapeamento 2346 determinar que os mapeamentos de versão de elemento de GUI estão disponíveis, então, a lógica de recuperação de mapeamento 2346 pode requisitar aqueles mapeamentos de versão de elemento de GUI (2810). A requisição pode ser feita para uma fonte local, tal como a memória 2304. De forma alternativa ou adicional, a requisição pode ser feita para um depósito de dados de elemento de GUI 138 através da lógica de comunicação 163. A requisição pode ser por mapeamentos específicos, tal como para mapeamentos de versão de elemento de GUI relevantes para o próximo nó. Alternativamente, a requisição pode ser por todos os mapeamentos de versão de elemento de GUI disponíveis. A lógica de recuperação de mapeamento 2346 pode receber os mapeamentos de versão de elemento de GUI em resposta à requisição (2812).

[00227] De forma alternativa ou adicional, a determinação quanto a se os mapeamentos de versão de elemento de GUI estão disponíveis, a requisição e a resposta podem ser combinadas em menos ações. Por exemplo, a lógica de recuperação de mapeamento 2346 pode apenas requisitar os mapeamentos. Uma resposta dos mapeamentos de versão de elemento de GUI pode confirmar que os mapeamentos estão disponíveis, enquanto uma resposta negativa ou nula pode confirmar que os mapeamentos não estão disponíveis.

[00228] A lógica de recuperação de mapeamento 2346 pode retornar, então, e a lógica de comparação de GAP 2342 então pode invocar a lógica de travessia de representação 2348. A lógica de travessia de representação 2348 pode atravessar para o próximo elemento de GUI, isto é, um elemento de GUI de origem, a partir do primeiro modelo de GUI de GAP (2814). No caso em que o modelo de diferença de GUI de base é recém criado, a travessia pode ser para o primeiro elemento de GUI. A travessia pode ser realizada com base na

representação do modelo de GUI de GAP dentro do modelo de diferença de GUI de base. O próximo nó a visitar pode ser determinado, como exemplos, em forma de primeiro na profundidade ou primeiro na largura.

[00229] A lógica de comparação de GAP 2342 então pode determinar se um mapeamento de versão de GUI existe para o elemento de GUI (2816). A lógica de comparação de GAP 2342 pode buscar nos mapeamentos de versão de elemento de GUI recuperados para determinar se o mapeamento existe. Se o mapeamento existir, então, a lógica de comparação de GAP 2342 poderá criar um campo de link no modelo de diferença de GUI de base (2818). O campo de link pode incluir um identificador de elemento de GUI, tal como um identificador de elemento de GUI de origem ou um identificador de elemento de GUI de destino. O campo de link também pode incluir um alias de GAP. A lógica de comparação de GAP 2342 pode criar um campo de link para apenas o elemento de GUI de origem. De forma alternativa ou adicional, a lógica de comparação de GAP pode criar um campo de link para o elemento de GUI de destino.

[00230] A lógica de travessia de representação 2348 pode determinar, então, se mais elementos de GUI de origem estão disponíveis (2820). Se mais elementos de GUI de origem não tiverem sido atravessados ainda, então, a lógica de travessia de representação 2348 circulará de volta e atravessará o próximo elemento de GUI de origem disponível (2814). Se não existirem elementos de GUI de origem disponíveis, a lógica de comparação de GAP 2342 poderá terminar.

[00231] Se nenhum mapeamento de versão de elemento de GUI existir ou os mapeamentos existirem, mas nenhum mapeamento existir para o elemento de GUI de origem, então, a lógica de comparação de GAP 2342 poderá invocar a lógica de análise ponderada 2350. A

lógica de análise ponderada 2350 pode recuperar pesos a partir de uma tabela de peso (2822). Por exemplo, a lógica de análise ponderada 2350 pode recuperar os pesos a partir de uma tabela de peso na memória 2304. De forma alternativa ou adicional, a lógica de análise ponderada 2350 pode requisitar e receber uma tabela de peso a partir da lógica de comunicação 163.

[00232] A lógica de travessia de representação 2348 então pode atravessar para o próximo elemento de GUI, isto é, um elemento de GUI de destino, no segundo modelo de GUI de GAP (2824). No caso em que nenhuma travessia prévia no segundo modelo de GUI de GAP foi feita para um dado elemento de GUI de origem, então, a lógica de travessia de representação 2348 pode atravessar para o primeiro elemento de GUI no segundo modelo de GUI de GAP. A travessia pode ser realizada com base no segundo modelo de GUI de GAP. De forma alternativa ou adicional, a travessia pode ser realizada com base na reprodução segundo modelo de GUI de GAP dentro do modelo de diferença de GUI de base. A travessia pode ser realizada usando-se uma técnica de travessia primeiro na profundidade, primeiro na largura ou outra.

[00233] A lógica de análise ponderada 2350 então pode obter características de elemento de GUI para o elemento de GUI de origem e o elemento de GUI de destino (2826). As características de elemento de GUI podem incluir características de elemento de GUI tais como tamanho, posição XY, cor, posição de janela, tipo de fonte, tamanho de fonte, estilo de borda, cor de fundo, cor de primeiro plano, apenas de leitura/apenas de escrita, ou qualquer outra característica de elemento de GUI. De forma alternativa ou adicional, as características de elemento de GUI podem incluir um Papel de camada de acessibilidade, HWND, Classe, Estilo, Estilo Estendido, número de filhos, nível em uma árvore ou hierarquia, Nome do elemento de GUI

ou outras propriedades de camada de acessibilidade atribuída. Estas características de elemento de GUI podem ser obtidas a partir dos primeiro e segundo modelos de GUI de GAP. De forma alternativa ou adicional, as características de elemento de GUI podem ser obtidas a partir do modelo de diferença de GUI de base.

[00234] A lógica de análise ponderada 2350 então pode determinar um valor de similaridade de elemento de GUI para o elemento de GUI de origem e o elemento de GUI de destino (2828). O valor de similaridade pode ser determinado de acordo com a fórmula a seguir:

$$V_s = \sum_{i=1}^{N} W_i \cdot P_i$$

[00235] onde V_s é o valor de similaridade, N é o número de características ou propriedades em relação às quais a similaridade está sendo medida, P_i é o valor atribuído às diferenças entre cada propriedade ou característica, e W_i é o peso característica para cada propriedade P_i. Como um exemplo:

$$V_s = W_R \cdot Role$$

[00236] onde *Role* pode ser 0 ou 1 dependendo de as características de Role entre os dois elementos de GUI serem diferentes ou as mesmas, respectivamente, e W_R pode ser o peso correspondente para Role. Ao peso para Role pode ser atribuído um valor indicativo da importância da combinação de Role entre os elementos de GUI. Por exemplo, o peso do Role pode ser muito grande em relação ao peso para outras características.

i. Como um outro exemplo:

$$V_s = W_R \cdot Role + W_c \cdot Class$$

[00237] onde *Class* pode ser uma contagem de quantos termos na propriedade Class combinam, dividido pelo número total de termos na propriedade Class, e W_c pode ser o peso correspondente para a Class. Por exemplo, uma característica de Class para um elemento de

GUI pode ser "WindowsForms10.LISTBOX.app4". Se a característica de Class para um elemento de GUI correspondente for "WindowsForms10.COMBOBOX.app.0.378734a", então devido ao fato de as características apenas combinarem em um único lugar de três ou cinco lugares, o valor de Class será "1/3" ou "1/5".

[00238] A lógica de comparação de GAP 2342 pode armazenar, então, em uma tabela de escore um escore com base no valor de similaridade (2830). De forma alternativa ou adicional, a lógica de comparação de GAP 2342 pode armazenar os identificadores de elemento de GUI para o elemento de GUI de origem e o elemento de GUI de destino juntamente com o escore na tabela de escore. A tabela de escore pode residir na memória 2304.

[00239] A lógica de comparação de GAP 2342 então pode determinar se a lógica de travessia de representação 2348 completou a travessia do segundo modelo de GUI de GAP (2832). Se a lógica de travessia de representação 2348 ainda tiver mais elementos de GUI de destino a atravessar, então, a lógica de travessia de representação 2348 circulará de volta para atravessar para o próximo elemento de destino (2824). Se a lógica de travessia de representação 2348 tiver completado a travessia dos elementos de GUI de destino, a lógica de comparação de GAP 2342 poderá invocar a lógica de construção de combinação 2352.

[00240] A lógica de construção de combinação 2352 pode analisar os valores de similaridade determinados pela lógica de análise ponderada 2350 ou os escores na tabela de escore (2834). A lógica de construção de combinação 2352 pode comparar os valores ou os escores em relação a um limite de similaridade para determinar se os valores ou escores se adéquam e/ou excedem ao limite de similaridade. De forma alternativa ou adicional, os valores ou escores podem ser comparados em relação a um limite de diferença para se

determinar se os valores ou escores estão em ou abaixo de um limite de diferença.

[00241] A lógica de construção de combinação 2352 pode determinar se os elementos de GUI combinam (2836). Esta determinação pode ocorrer quando os valores ou escores excederem ao limite de similaridade. De forma alternativa ou adicional, esta determinação pode ocorrer quando os valores ou escores não estiverem abaixo de um limite de diferença.

[00242] Se existir uma combinação, então, a lógica de comparação de GAP 2342 poderá criar um campo de link no modelo de diferença de GUI de base (2838). O campo de link pode incluir um identificador de elemento de GUI, tal como um identificador de elemento de GUI de origem ou um identificador de elemento de GUI de destino. O campo de link também pode incluir um alias de GAP. A lógica de comparação de GAP 2342 pode criar um campo de link para apenas o elemento de GUI de origem. De forma alternativa ou adicional, a lógica de comparação de GAP pode criar um campo de link para o elemento de GUI de destino.

[00243] Após a lógica de comparação de GAP 2342 criar um campo de link, ou se nenhuma combinação existir, então, a lógica de construção de combinação 2352 pode determinar se mais escores ou valores de similaridade precisam ser analisados (2840). Esta determinação pode depender de a tabela de escore ainda incluir escores que não foram analisados. Se a lógica de comparação de GAP 2342 determinar que mais escores precisam ser analisados, a lógica de construção de combinação 2352 circulará de volta para analisar o próximo escore na tabela de escore (2834). Se nenhum escore não analisado existir na tabela de escore, a lógica de construção de combinação 2352 poderá retornar, e a lógica de comparação de GAP 2342 poderá circular de volta para determinar se

quaisquer elementos de GUI de origem a mais permanecem (2820). De forma alternativa ou adicional, a lógica de comparação de GAP 2342 pode comunicar quaisquer campos de link criados pela lógica de construção de combinação 2352 para o depósito de dados de elemento de GUI para armazenamento como um mapeamento de versão de elemento de GUI de GAP. A comunicação pode usar um formato de mensagem de mapeamento de versão de elemento de GUI, conforme descrito na Figura 6.

[00244] Em uma outra implementação, a lógica de comparação de GAP 2342 pode executar uma comparação de esquema para determinar diferenças entre a GUI de GAP atual e a GUI de GAP subsequente. Dada uma representação de esquema (por exemplo, uma representação de esquema em XML) da GUI de GAP atual e da GUI de GAP subsequente, a lógica de comparação de GAP 2342 pode comparar os respectivos esquemas. Se estes esquemas forem "iguais", então, a versão de GAP atual e a versão de GAP subsequente serão as mesmas. Caso contrário, a versão de GAP atual e a versão de GAP subsequente serão diferentes e a lógica de comparação de GAP 2342 encontrará as diferenças.

[00245] Por exemplo, os esquemas em XML podem ser registrados no formato XML e cada esquema pode ter a raiz especificada com o elemento <schema>. Elementos de dados podem ser especificados com os tags <element> e <attribute>. Cada elemento de dados pode ser definido por seu nome e por seu tipo. Os elementos podem ser de tipos simples ou complexos. Os tipos de elemento complexos suportam elementos aninhados, enquanto os tipos simples são atributos e elementos de tipos básicos.

[00246] Estendendo o exemplo, os elementos podem ter dois tipos de restrições. Em primeiro lugar, os valores dos elementos podem ser restritos. O segundo tipo de restrições especifica limites quanto ao

número de vezes que um elemento específico pode ocorrer como um filho de um elemento. Estes limites podem ser especificados com os atributos minOccurs e maxOccurs do tag <element> para representação do número mínimo e do máximo de ocorrências. Os elementos podem ser agrupados em uma sequência se eles forem filhos do mesmo elemento pai. A cada elemento ou atributo em uma sequência pode ser atribuído um número de sequência inteiro positivo único. Este número pode ser usado para se acessarem elementos ou atributos, ao invés de se usarem seus nomes.

[00247] Os esquemas podem ser representados usando-se gráficos. Seja T os conjuntos finitos de nomes de tipo e F de nomes de elemento e atributo (rótulos) e símbolos distintos $\alpha \in F$ e $\beta \in T$. Os gráficos de esquema são gráficos dirigidos G (V, E, L), de modo que:

1) $V \subseteq T$, os nós são nomes de tipo ou β se o nome de tipo de dados não for conhecido;

2) $L \subseteq F$, bordas são rotuladas por nomes de elemento ou de atributo ou α se o nome não for conhecido;

3) $E \subseteq L \times V \times V$, bordas são produtos vetoriais de rótulos e nós. Se <l, vk, vm> $\in$ E, então vk -> (l) -> vm. Os nós vm são chamados de filhos do nó vk. Se um elemento não tiver filhos, então, seu nó correspondente em um gráfico de esquema terá uma coleção vazia de nós filhos;

4) Os limites para os elementos são especificados com subscritos e sobrescritos para rótulos designando estes elementos. Os subscritos são usados para especificação de limites definidos pelo atributo minOccurs, e sobrescritos designam os limites especificados pelo atributo maxOccurs;

5) Cada gráfico tem um nó especial rotulado root $\in$ V, onde root representa uma coleção de elementos de raiz. Um esquema vazio tem um nó de raiz único e nenhuma borda;

6) O tag de XML <complexType> especifica que um elemento tem um tipo complexo, e não é representado no gráfico.

[00248] Um caminho em um gráfico de esquema pode ser uma se de rótulos PG = <l1, l2, ... ln>, onde vk -> (ln) -> vm para vm em V e 1 <= u <= n. O símbolo β pode ser usado ao invés de um rótulo em um caminho, se um elemento for navegado por seu número de sequência. Function type:v->s retorna o tipo s $\in$ T do nó v $\in$ V. Function max:label(u, l) -> u retorna o limite superior u, ou ∞ se o limite superior não for especificado, e function min:label(u, l) -> l retorna o limite inferior l, ou zero, se o limite inferior não for especificado.

[00249] Uma vez que os GAPs sejam modelados usando-se esquemas em XML, estes esquemas podem ser comparados usando-se uma simulação para computação de mudanças entre os GAPs correspondentes. Isto é, se o esquema do novo GAP for o mesmo que o esquema da liberação prévia do GAP, ou os tipos de seus objetos de GUI forem subsumidos pelos tipos do objeto de GUI correspondente do GAP prévio, então, estes esquemas poderão ser considerados idênticos. Caso contrário, a lógica de comparação de GAP 2342 poderá emitir um aviso e os objetos de GUI com os tipos de modificação associados serão reportados.

[00250] Para essa finalidade, a lógica de comparação de GAP 2342 pode implementar uma técnica de bissimulação para comparação de esquemas. A Figura 32 mostra um exemplo das propriedades de bissimulação 3200 que uma lógica de comparação de GAP 2342 pode empregar, incluindo uma primeira propriedade de bissimulação 3202, uma segunda propriedade de 3204, uma terceira propriedade de bissimulação 3206 e uma quarta propriedade de bissimulação 3208.

[00251] A bissimulação pode ser uma relação binária entre os nós de dois gráficos g1, g2 $\in$ G, escrita como x $\sim$ y, x, y $\in$ V, satisfazendo às propriedades de bissimulação 3202 - 3208.

[00252] A lógica de comparação de GAP 2342 pode considerar dois gráficos finitos g1, g2 $\in$ G iguais, se existir uma bissimulação de g1 para g2. Um gráfico é bissimilar para seu desdobramento infinito. A lógica de comparação de GAP 2342 pode computar a bissimulação de dois gráficos começando com uma seleção dos nós de raiz e aplicando as propriedades de bissimulação 3202 – 3208. A lógica de comparação de GAP 2342 busca uma relação (x, y) entre os nós x e y em um gráfico que falha em satisfazer às propriedades de bissimulação 3202 – 3208. Quando uma relação (x, y) como essa é encontrada, então, a lógica de comparação de GAP 2342 determinará que os gráficos não são iguais e a bissimulação poderá parar.

[00253] Por exemplo, considere o esquema atual 3210 e o esquema subsequente 3212 na Figura 32. A lógica de comparação de GAP 2342 aplica uma bissimulação para determinar se dois esquemas 3210 e 3212 são equivalentes. O esquema 3210 descreve dados em XML que modelam uma tela de GUI atual, e o esquema 3212 modela uma versão modificada da tela de GUI atual. A lógica de comparador 2342 pode determinar que se o esquema 3212 for equivalente ao esquema 3210, então, as telas de GUI serão as mesmas.

[00254] A lógica de comparação de GAP 2342 seleciona os nós de raiz 3214 e 3216 em ambos os esquemas 3210 e 3212 que satisfazem à primeira propriedade de bissimulação 3202 e à segunda propriedade de bissimulação 3204. A lógica de comparação de GAP 2342 então pode selecionar a relação α -> (author1) -> α 3222 para o esquema 3210 e a relação α -> (author1) -> α 3224 para o esquema 3212. A lógica de comparação de GAP 2342 determina que a terceira propriedade de bissimulação 3206 e a quarta propriedade de bissimulação 3208 são violadas. Em particular, a lógica de comparação de GAP 2342 determina que a relação ofendendo 'author' é marcada como potencialmente apagada no esquema 3212, uma

diferença do esquema 3210. Assim, os esquemas 3210 e 3212 não são iguais.

[00255] A Figura 29 mostra um visor 2900 e uma interface de limite de comparador de GUI 2902. A lógica de modelo de diferença de GUI 2380 pode exibir a interface 2902 em resposta a uma entrada de operador. Por exemplo, o operador pode selecionar um item de configuração de opção a partir de um menu gerado pela lógica de modelo de diferença de GUI 2380. Em resposta, a lógica de modelo de diferença de GUI 2380 exibe a interface 2902. A exibição pode ser provida localmente, por exemplo, através do visor 2306, ou remotamente, por exemplo, via o terminal 2318.

[00256] No exemplo mostrado na Figura 29, a interface 2902 inclui um cursor deslizante 2904 que seleciona um valor entre 0 e 1. Qualquer outra interface ou faixa de valor pode ser provida para o operador. A lógica de modelo de diferença de GUI 2380 pode regular o limite de diferença com base no valor do cursor deslizante 2904. O valor 9 representa que essencialmente nenhuma similaridade ou uma similaridade limitada é necessária para se encontrar uma combinação entre os elementos de GUI. O valor 1 representa que uma similaridade muito próxima ou exata é necessária para se encontrar uma combinação entre elementos de GUI. Essa similaridade pode ser encontrada em mapeamentos manuais, por exemplo, conforme especificado em um campo de nível de confiança alto 811 (por exemplo, "100M") conforme recebido a partir do depósito de metadados. Contudo, um nível de confiança muito alto também pode ser obtido através da análise de cálculo de peso automatizada descrita acima, e a lógica de modelo de diferença de GUI 2380 em algumas implementações pode aceitar um mapeamento de versão manual como correto, independentemente do nível de confiança associado.

[00257] A Figura 29 também mostra uma porção 2906 da versão de

GAP atual 2326 e uma porção 2908 da versão de GAP subsequente 2328. Além de permitir que o operador regule o limite de usando a interface 2902, a lógica de modelo de diferença de GUI 2380 ainda pode incluir uma lógica de visualização 2378. A lógica de visualização 2378 pode exibir elementos na versão de GAP atual e na versão de GAP subsequente que combinem, conforme determinado pela análise de comparação ponderada, pelo limite de similaridade e pelos mapeamentos de versão. Os elementos de GUI combinando podem ser destacados com uma borda de um padrão em particular ou cor, ou de outras maneiras, e bordas, cores e outros recursos diferentes podem ser usados para cada combinação.

[00258] Na Figura 29, a lógica de visualização 2378 destaca os elementos de GUI combinando com base no limite de similaridade regulado através da interface 2902. O limite de similaridade é muito alto. No exemplo mostrado na Figura 29, a lógica de visualização 2378 destaca os elementos de caixa de texto 2910, 2912, 2914, 2916 e 2918 na porção 2906 da versão de GAP atual 2326 que combinam, respectivamente, com os elementos de texto 2920, 2922, 2924, 2926 e 2928 na porção 2908 da versão de GAP subsequente 2328. Os elementos de caixa de texto 2190 – 2928 têm poucas mudanças ou nenhuma nas suas características entre as versões de GAP subsequentes. O elemento de caixa de texto 2930 e o elemento de caixa de combinação 2932 permanecem não destacados, contudo, por que suas características diferem até uma grande extensão, e a análise de comparação ponderada não determina um valor de similaridade de elemento de GUI que exceda ao limite de similaridade.

[00259] Na Figura 30, o visor 3000 mostra que o cursor deslizante 2904 foi ajustado para um limite de similaridade mais baixo. A lógica de visualização 2378 destaca os elementos de GUI de combinação com base no limite de similaridade mais baixo regulado através da

interface 2902. No exemplo mostrado na Figura 30, a lógica de visualização 2378 destaca, como antes, os elementos de caixa de texto 2910, 2912, 2914, 2916 e 2918 na porção 2906 da versão de GAP atual 2326 que combinam, respectivamente, com os elementos de texto 2920, 2922, 2924, 2926 e 2928 na porção 2908 da versão de GAP subsequente 2328.

[00260] Contudo, a lógica de visualização 2378 também destaca o elemento de caixa de texto 2930 e o elemento de caixa de combinação 2932. Embora as características do elemento de caixa de texto 2930 e do elemento de caixa de combinação 2932 difiram até uma certa extensão, a análise de comparação ponderada realmente obtém um valor de similaridade de elemento de GUI que excede ao limite de similaridade. Assim sendo, a lógica de visualização 2378 destaca os elementos 2930 e 2932.

[00261] A Figura 31 mostra um fluxograma 3100 par artigo absorvente lógica de visualização 2378. A lógica de visualização 2378 também pode exibir uma interface de limite de comparador de GUI 2902 (3106). A lógica de visualização 2378 pode regular o limite de similaridade com base no valor escolhido através da interface de limite de comparador de GUI 2902 (3108). Dado o limite de similaridade, a lógica de visualização 2378 pode chamar a versão de GAP atual 2326 e a versão de GAP subsequente 2328 (3110). Alternativamente, a lógica de visualização 2378 pode executar uma análise de comparação (por exemplo, a análise de comparação ponderada descrita acima) para determinar um ou mais elementos de GUI na versão de GAP subsequente 2328 que combinem com qualquer elemento em particular na versão de GAP atual. A lógica de visualização 2378 pode aceitar uma seleção de elemento a partir do operador, que especifique um ou mais elementos de GUI em particular de interesse em qualquer versão de GAP, e encontrar os elementos de

GUI combinando na outra versão de GAP. Alternativamente, a lógica de visualização 2378 pode considerar cada elemento de GUI na versão de GAP atual 2326 e encontrar os elementos de GUI combinando na versão de GAP subsequente 2328.

[00262] A lógica de visualização 2378 destaca os elementos de GUI combinando na versão de GAP atual 2326 e na versão de GAP subsequente 2328 (3112). Para essa finalidade, a lógica de visualização 2378 pode emitir comandos para o proxy para destacar quaisquer elementos de GUI em particular. Se houver mais elementos de GUI a considerar, a lógica de visualização 2378 tentará encontrar combinações adicionais.

[00263] Em qualquer momento, a lógica de visualização 2378 pode checar para determinar se a interface de limite de comparador de GUI 2902 mudou (por exemplo, o operador mudou a posição do cursor deslizante para selecionar um novo valor de limite). A lógica de visualização 2378 também pode checar, em qualquer momento, se o operador desejou rever GAPs diferentes. Caso assim seja, a lógica de visualização 2378 obtém novas seleções de GAP (3114). A lógica de visualização então exibe os GAPs e a interface de limite de comparador de GUI e prossegue conforme citado acima.

[00264] A Figura 1 mostra um analisador de script com uma arquitetura de guia de mudança (SAA) 108. Embora descrições detalhadas dos recursos da SAA 108 sejam providas adicionalmente abaixo, uma breve introdução da SAA 108 será apresentada, primeiramente. A SAA 108 recebe um modelo de diferença de GUI 162 que especifica diferenças de elemento de GUI entre uma versão de GAP atual 150 e uma versão de GAP subsequente 152. O modelo de diferença de GUI 162 pode ser representado como um esquema em XML. Em uma outra implementação, os modelos de árvore de GAP atual e subsequente, bem como o modelo de diferença de GUI 162

são implementados como modelos relacionais armazenados em um banco de dados. A SAA 108 emprega uma interface 190 para receber entradas e se comunicar com vários componentes, incluindo um depósito de metadados de elemento de GUI 138. O depósito de metadados de elemento de GUI 138 pode prover uma informação detalhada referente aos elementos de GUI representados no modelo de diferença de GUI 162, o GAP atual 150 e o GAP subsequente 152. Em uma implementação, a SAA 108 inclui um analisador gramatical de script 166 que analisa gramaticalmente um script de teste atual 164 para a obtenção de uma representação intermediária do script de teste atual 164. A representação intermediária pode ser uma árvore de sintaxe abstrata (AST) 168 ou uma outra representação do script de teste atual 164. A SAA 108 emprega um analisador de script 170 que analisa a AST 168 e invoca uma consulta de depósito de objeto 172 em relação a um depósito de objeto 174, para a localização das propriedades de elementos de GUI identificados pela AST 168. O analisador de script 170 usa o depósito de metadados de elemento de GUI 138, o depósito de objeto 174, um motor de satisfação de restrição 188 e regras de mudança de script de elemento de GUI 194 para a localização de entradas de diferença de elemento de GUI válidas, discutidas adicionalmente abaixo. Em uma implementação, o analisador de script 170 usa uma lógica de regras de classe de GUI para favorecer a análise, discutido em maiores detalhes abaixo. O analisador de script 170 extrai um script de teste transformado 178, um guia de mudança 180 e especificadores de mudança de GAP 184 com base na análise realizada no modelo de diferença de GUI 162 e na AST 168.

[00265] A Figura 33 mostra uma GUI de uma versão de GAP atual 150. A Tabela 6 mostra uma representação de modelo de árvore de GAP atual da GUI da versão de GAP atual 150. O modelo de árvore

de GAP atual mostrado na Tabela 6 especifica os elementos de GUI e os atributos dos elementos de GUI da versão de GAP atual 150. A Tabela 6 ilustra que o modelo de árvore de GAP atual suporta elementos de GUI que incluem elementos de GUI aninhados. Por exemplo, a janela StateList 3302, mostrada na Figura 33, corresponde a GUI Element Alias StateList na linha 11 (L11) da Tabela 6 e elementos de GUI aninhados SaveFile, Exit, SaveChange, FileOpen, a caixa de lista School, mostrados nas linhas 18-21 e 59 da Tabela 6, correspondem, respectivamente, a Save File 3304, Close Form 3306, Save Change 3308, Open File 3310 e à caixa de lista School 3316 da Figura 33. Em uma implementação, o modelo de diferença de GUI 162 resulta de uma comparação do modelo de árvore de GAP atual, conforme mostrado na Tabela 6 e um modelo de árvore de GAP subsequente ilustrado abaixo na Tabela 8.

[00266] Um GAP, os elementos de GUI do GAP e os valores dos elementos de GUI definem estados para o GAP. Os modelos de árvore de GAP atual e subsequente capturam os estados das versões de GAP atual e subsequente (por exemplo, 150 e 152), respectivamente. Em uma implementação, os estados de GAP são identificados por números de sequência e alias, bem como outros atributos. Por exemplo, a linha 1 da Tabela 6 ilustra um 'state' que tem um SeqNumber com um valor de 0. O SeqNumber representa um número de sequência único da versão de GAP atual. Ao estado é dado o nome State_0_3556. Os atributos Alias e Processid representam o alias da versão de GAP atual 150 e o identificador de processo de instância para a versão de GAP atual 150, respectivamente. Lembre que a Tabela 6 e a Tabela 8 ilustram que os modelos de árvore de GAP atual e subsequente suportam elementos de GUI que incluem elementos de GUI aninhados. Embora múltiplos elementos de GUI possam usar um Alias idêntico (por exemplo, StateList, conforme ilustrado na Tabela 6

nas linhas 2 e 11), os elementos de GUI são adicionalmente distinguidos pelo atributo UniqueID (por exemplo, 0x0 e 0x12, conforme mostrado nas linhas 3 e 12 da Tabela 6).

Tabela 6 – Modelo de árvore de GAP atual

```
- <State SeqNumber="0" Name="State_0_3556" Alias="University Directory0" ProcessId="3556">
 - <GUIElement Alias="StateList">
      <UniqueID>0x0</UniqueID>
      <HWND>0x170a64</HWND>
      <Location x="87" y="66" width="792" height="672" />
      <Class>WindowsForms10.Window.8.app4</Class>
      <Style>0x16cf0000</Style>
      <ExStyle>0xc0050900</ExStyle>

      + <GUIElement Alias="System">

      + <GUIElement Alias="NAMELESS">

L11   - <GUIElement Alias="StateList">
      <UniqueID>0x12</UniqueID> fs
      <HWND>0x170a64</HWND>
      <Location x="117" y="70" width="784" height="638" />
      <Class>WindowsForms10.Window.8.app4</Class>
      <Style>0x16cf0000</Style>
      <ExStyle>0xc0050900</ExStyle>
L18   + <GUIElement Alias="SaveFile">
L19   + <GUIElement Alias="Exit">
L20   + <GUIElement Alias="SaveChange">
L21   + <GUIElement Alias="FileOpen">
      + <GUIElement Alias="Location">
      + <GUIElement Alias="AcademicEmph">
       + <GUIElement Alias="QolScale">
      + <GUIElement Alias="SocialScale">
      + <GUIElement Alias="AcadScale">
```

```
    + <GUIElement Alias="EnrolledPerc">
    + <GUIElement Alias="AdmittancePerc">
    + <GUIElement Alias="NumApps">
    + <GUIElement Alias="FinancialAid">
    + <GUIElement Alias="Expense">
    + <GUIElement Alias="SATMath">
    + <GUIElement Alias="SATVerbal">
    + <GUIElement Alias="SFRatio">
    + <GUIElement Alias="MFRatio">
    + <GUIElement Alias="NumStudents">
    + <GUIElement Alias="Control">
    + <GUIElement Alias="State">
    + <GUIElement Alias="School">
    + <GUIElement Alias="Location">
    + <GUIElement Alias="Academic Emphasis">
    + <GUIElement Alias="Quality of Life Scale (1-5)">
    + <GUIElement Alias="Social Scale (1-5)">
    + <GUIElement Alias="Academics Scale (1-5)">
    + <GUIElement Alias="Enrolled %">
    + <GUIElement Alias="Admittance %">
    + <GUIElement Alias="# Applicants (1000)">
    + <GUIElement Alias="Financial Aid %">
    + <GUIElement Alias="Expenses (1000$)">
    + <GUIElement Alias="SAT:math">
    + <GUIElement Alias="Student/Faculty Ratio">
    + <GUIElement Alias="SAT:verbal">
    + <GUIElement Alias="Male/Female Ratio">
    + <GUIElement Alias="Number of Students (1000)">
    + <GUIElement Alias="Control">
    + <GUIElement Alias="State">
    + <GUIElement Alias="SelectSchoolBtn">
    + <GUIElement Alias="School List">
L59 + <GUIElement Alias="SchoolListbox">
    + <GUIElement Alias="SelectStateBtn">
```

```
      + <GUIElement Alias="State List">
L62   + <GUIElement Alias="StateListbox">
  </GUIElement>
 </GUIElement>

 </State>
```

[00267] O elemento de GUI StateListBox mostrado na Tabela 6 na linha 62 corresponde à caixa de lista State 3312 mostrada na Figura 33. A Figura 33 mostra uma barra de navegação horizontal 3314 como um recurso da caixa de lista State 3312. A Tabela 7 mostra alguns dos atributos da caixa de lista State 3312 que podem ser refletidos no modelo de diferença de GUI 162 como resultado de uma comparação entre o modelo de árvore de GAP atual e o modelo de árvore de GAP subsequente mostrado na Tabela 8.

Tabela 7 – Esquema de elemento de GUI de StateListbox de GAP atual

```
- <GUIElement Alias="StateListbox">
   <UniqueID>0x407</UniqueID>
   <HWND>0x90b58</HWND>
   <Location x="173" y="86" width="368" height="274" />
   <Class>WindowsForms10.LISTBOX.app4</Class>
   <Style>0x56110ac1</Style>
   <ExStyle>0xc0000a00</ExStyle>

  - <GUIElement Alias="StateListbox">
   <UniqueID>0x410</UniqueID>
   <HWND>0x90b58</HWND>
   <Location x="175" y="88" width="364" height="253" />
   <Class>WindowsForms10.LISTBOX.app4</Class>
   <Style>0x56110ac1</Style>
   <ExStyle>0xc0000a00</ExStyle>
```

- <Values>
<Value SeqNumber="0">Alabama</Value>
<Value SeqNumber="1">Alaska</Value>
<Value SeqNumber="2">Arizona</Value>
<Value SeqNumber="3">Arkansas</Value>
<Value SeqNumber="4">California</Value>
<Value SeqNumber="5">Colorado</Value>
<Value SeqNumber="6">Connecticut</Value>
<Value SeqNumber="7">Delaware</Value>
<Value SeqNumber="8">District of Columbia</Value>
<Value SeqNumber="9">Florida</Value>
<Value SeqNumber="10">Georgia</Value>
<Value SeqNumber="11">Hawaii</Value>
<Value SeqNumber="12">Idaho</Value>
<Value SeqNumber="13">Illinois</Value>
<Value SeqNumber="14">Indiana</Value>
<Value SeqNumber="15">Iowa</Value>
<Value SeqNumber="16">Kansas</Value>
<Value SeqNumber="17">Kentucky</Value>
<Value SeqNumber="18">Louisiana</Value>
<Value SeqNumber="19">Maine</Value>
<Value SeqNumber="20">Maryland</Value>
<Value SeqNumber="21">Massachusetts</Value>
<Value SeqNumber="22">Michigan</Value>
<Value SeqNumber="23">Minnesota</Value>
<Value SeqNumber="24">Mississippi</Value>
<Value SeqNumber="25">Missouri</Value>
<Value SeqNumber="26">Montana</Value>
<Value SeqNumber="27">Nebraska</Value>
<Value SeqNumber="28">Nevada</Value>
<Value SeqNumber="29">New Hampshire</Value>
<Value SeqNumber="30">New Jersey</Value>

```
<Value SeqNumber="31">New Mexico</Value>
<Value SeqNumber="32">New York</Value>
<Value SeqNumber="33">North Carolina</Value>
<Value SeqNumber="34">North Dakota</Value>
<Value SeqNumber="35">Ohio</Value>
<Value SeqNumber="36">Oklahoma</Value>
<Value SeqNumber="37">Oregon</Value>
<Value SeqNumber="38">Pennsylvania</Value>
<Value SeqNumber="39">Rhode Island</Value>
<Value SeqNumber="40">South Carolina</Value>
<Value SeqNumber="41">South Dakota</Value>
<Value SeqNumber="42">Tennessee</Value>
<Value SeqNumber="43">Texas</Value>
<Value SeqNumber="44">Utah</Value>
<Value SeqNumber="45">Vermont</Value>
<Value SeqNumber="46">Virginia</Value>
<Value SeqNumber="47">Washington</Value>
<Value SeqNumber="48">West Virginia</Value>
<Value SeqNumber="49">Wisconsin</Value>
<Value SeqNumber="50">Wyoming</Value>
</Values>

</GUIElement>

+ <GUIElement Alias="Horizontal">

</GUIElement>
```

[00268] A Figura 34 mostra a GUI de uma versão de GAP subsequente 152. A Tabela 8 ilustra uma representação de modelo de árvore de GAP subsequente da versão de GAP subsequente 152. O modelo de árvore de GAP subsequente mostrado na Tabela 8 inclui os elementos de GUI e os atributos dos elementos de GUI. Por exemplo, o objeto de GUI de janela School 3402 mostrado na Figura 34

corresponde ao Alias de Elemento de GUI "School" mostrado na linha 11 (L11) da Tabela 8 e os elementos de GUI aninhados StateListBox e SchoolCombobox mostrados nas linhas 23 e 24 da Tabela 8 correspondem, respectivamente, à caixa de lista State 3404 e à caixa de combinação School 3406 da Figura 34. Em uma implementação, o modelo de diferença de GUI 162 resulta de uma comparação entre o modelo de árvore de GAP atual, conforme mostrado na Tabela 6 e o modelo de árvore de GAP subsequente, conforme mostrado na Tabela 8.

Tabela 8 – Modelo de árvore de GAP subsequente

- <State SeqNumber="**0**" Name="**State_0_3068**" Alias="**University Directory1**" ProcessId="**3068**">
- <GUIElement Alias="**School**">
<UniqueID>**0x0**</UniqueID>
<HWND>**0x80b8**</HWND>
<Location x="**116**" y="**88**" width="**915**" height="**594**" />
<Class>**WindowsForms10.Window.8.app.0.378734a**</Class>
<Style>**0x16cf0000**</Style>
<ExStyle>**0xc0050900**</ExStyle>

+ <GUIElement Alias="**System**">
+ <GUIElement Alias="**NAMELESS**">

L11 - <GUIElement Alias="**School**">
<UniqueID>**0x12**</UniqueID>
<HWND>**0x80b8**</HWND>
<Location x="**146**" y="**92**" width="**907**" height="**560**" />
<Class>**WindowsForms10.Window.8.app.0.378734a**</Class>
<Style>**0x16cf0000**</Style>
<ExStyle>**0xc0050900**</ExStyle>

L18 + <GUIElement Alias="**menuStrip1**">
+ <GUIElement Alias="**States List**">
+ <GUIElement Alias="**School List**">
+ <GUIElement Alias="**SelectStateIButton**">
+ <GUIElement Alias="**SelectSchoolButton**">
L23 + <GUIElement Alias="**StateListbox**">
L24 + <GUIElement Alias="**SchoolCombobox**">
+ <GUIElement Alias="**School**">
+ <GUIElement Alias="**state**">
+ <GUIElement Alias="**State**">
+ <GUIElement Alias="**location**">
+ <GUIElement Alias="**Location**">
+ <GUIElement Alias="**control**">
+ <GUIElement Alias="**Control**">
+ <GUIElement Alias="**Number of Students (1000)**">
+ <GUIElement Alias="**NumStudents**">
+ <GUIElement Alias="**Male/Female Ratio**">
+ <GUIElement Alias="**GenderRatio**">
+ <GUIElement Alias="**Student/Faculty Ratio**">
+ <GUIElement Alias="**SFRatio**">
+ <GUIElement Alias="**SAT Verbal**">
+ <GUIElement Alias="**SATVerbal**">
+ <GUIElement Alias="**SAT Math**">
+ <GUIElement Alias="**SATMath**">
+ <GUIElement Alias="**Number of Applicants**">
+ <GUIElement Alias="**NumApps**">
+ <GUIElement Alias="**Percent of Admittance**">
+ <GUIElement Alias="**PercAdmit**">
+ <GUIElement Alias="**Percent Enrolled**">
+ <GUIElement Alias="**Percent Enrolled**">
+ <GUIElement Alias="**Academics (1-5)**">
+ <GUIElement Alias="**Academics**">

```
+ <GUIElement Alias="Social (1-5)">
+ <GUIElement Alias="Social">
+ <GUIElement Alias="Quality of Life (1-5)">
+ <GUIElement Alias="QoLife">
+ <GUIElement Alias="Academic Emphasis">
+ <GUIElement Alias="AcadEmphasis">
+ <GUIElement Alias="Expenses">
+ <GUIElement Alias="Expense">
+ <GUIElement Alias="Financial Aid">
+ <GUIElement Alias="FinancialAid">

</GUIElement>
</GUIElement>
</State>
```

[00269] O esquema de elemento de GUI menuStrip1 mostrado na Tabela 8 na linha 18 corresponde ao objeto de GUI WinObject GUI 'menu strip' 3408 mostrado na Figura 34. O esquema de elemento de GUI menuStrip1 de GAP subsequente mostrado na Tabela 9 ilustra a entrada plena na linha 18 na Tabela 8 e indica que a menu strip 3408 inclui um elemento de GUI aninhado File menu que inclui os elementos de GUI aninhados OpenFile, SaveFile, SaveAs, e Exit, mostrados nas linhas 15 da Tabela 9, e 22-25, respectivamente.

Tabela 9 – Esquema de elemento de GUI menuStrip1 de GAP subsequente

- <GUIElement Alias="**menuStrip1**">
<UniqueID>**0x13**</UniqueID>
<HWND>**0xa0e62**</HWND>
<Location x="**146**" y="**92**" width="**907**" height="**24**" />
<Class>**WindowsForms10.Window.8.app.0.378734a**</Class>
<Style>**0x56000000**</Style>
<ExStyle>**0xc0010800**</ExStyle>

- <GUIElement Alias="**menuStrip1**">
<UniqueID>**0x1c**</UniqueID>
<HWND>**0xa0e62**</HWND>
<Location x="**146**" y="**92**" width="**907**" height="**24**" />
<Class>**WindowsForms10.Window.8.app.0.378734a**</Class>
<Style>**0x56000000**</Style>
<ExStyle>**0xc0010800**</ExStyle>

L15 - <GUIElement Alias="**FileMenu**">
<UniqueID>**0x1d**</UniqueID>
<HWND>**0xa0e62**</HWND>
<Location x="**148**" y="**98**" width="**35**" height="**20**" />
<Class>**WindowsForms10.Window.8.app.0.378734a**</Class>
<Style>**0x56000000**</Style>
<ExStyle>**0xc0010800**</ExStyle>

L22 + <GUIElement Alias="**OpenFile**">
L23 + <GUIElement Alias="**SaveFile**">
L24 + <GUIElement Alias="**SaveAsFile**">
L25 + <GUIElement Alias="**Exit**">
</GUIElement>

```
</GUIElement>

</GUIElement>
```

[00270] A Figura 35 ilustra uma vista lado a lado do GAP atual e do subsequente, que ajuda a ilustrar similaridades de elemento de GUI, bem como diferenças, entre versões de GAP sucessivas. Na vista 3500, há vários objetos de GUI que têm a mesma funcionalidade desejada entre versões de GAP sucessivas, embora aspectos do objeto de GUI possam parecer diferentes entre versões de GAP sucessivas.

[00271] Por exemplo, com referência à Figura 35, diferenças entre a versão de GAP atual 150 e a versão de GAP subsequente 152 incluem que a janela StateList 3302, a caixa de lista State 3312, o campo Academic Emphasis 3502, e o campo Quality of Life Scale (1-5) 3504 na versão de GAP atual 150 são respectivamente representados pela janela School 3402, pela caixa de lista State 3404, pelo campo Academic Emphasis 3506, e pelo campo Quality of Life (1-5) 3508 na versão de GAP subsequente 152. Em um outro exemplo, considere os objetos de GUI Save File 3304, Close Form 3306, Save Change 3308 e Open File 3310 implementados na versão de GAP atual 150 que foram implementados na versão de GAP subsequente 152 como objetos de GUI filhos do File 3510, o qual é um objeto de GUI filho do objeto de GUI menu strip 3408.

[00272] Pode ser desafiador localizar diferenças entre elementos de GUI de GAPs. Por exemplo, não é prontamente evidente que a caixa de lista School 3316 e a caixa de combinação School 3406 têm por significado ter a mesma funcionalidade ou uma similar entre versões de GAP sucessivas. Como um outro exemplo, o WinObject "Select School" 3318 na versão de GAP atual 150 foi removido na localização 3512 da versão de GAP subsequente 152. O modelo de diferença de

GUI 162 inclui as entradas de diferença de elemento de GUI que listam características de elementos de GUI, para aqueles elementos de GUI que combinam entre a versão de GAP atual e a versão de GAP subsequente, mas que diferem no caráter entre a versão de GAP atual e a versão de GAP subsequente. As entradas de diferença de elemento de GUI guiarão a análise de script conforme descrito em maiores detalhes abaixo.

[00273] A Figura 36 mostra uma entrada de diferença de elemento de GUI de exemplo 3604. A Figura 36 ilustra um modelo de diferença de GUI 162 obtido pela comparação de um modelo de árvore de GAP atual, conforme mostrado na Tabela 6 e um modelo de árvore de GAP subsequente, conforme mostrado na Tabela 8. Em uma implementação, o modelo de diferença de GUI 162 é implementado como um esquema em XML que especifica cada diferença de elemento de GUI entre versões de GAP sucessivas com uma entrada de diferença de elemento de GUI correspondente. Em uma outra implementação, o modelo de diferença de GUI 162 aninha entradas de diferença de elemento de GUI para indicar elementos de GUI pais e filhos, e o nível no qual uma entrada de diferença de elemento de GUI é aninhada indica quão distante o elemento de GUI está de um elemento de GUI pai (raiz) em um caminho de navegação.

[00274] Em uma implementação, o modelo de diferença de GUI 162 omite uma entrada de diferença de elemento de GUI para objetos de GUI que tenham sido apagados entre versões de GAP sucessivas. Cada entrada de diferença de elemento de GUI representando um objeto de GUI que tenha sido modificado ou adicionado entre versões de GAP sucessivas inclui um tag 'Version' que tem um valor de 0 ou 1, como exemplo. Em outras palavras, uma entrada de diferença de elemento de GUI que não inclui um tag Version indica que o objeto de GUI não foi modificado entre versões de GAP sucessivas. Os valores

0 e 1 de Version indicam se os elementos filhos da Version representam as propriedades do objeto de GUI na versão de GAP atual 150 ou na versão de GAP subsequente 152, respectivamente. Por exemplo, a entrada de diferença de GUI 3604 mostrada na Figura 36 indica na linha 11 que o valor de caixa de lista de StateListbox para SeqNumber="8" implementado na versão de GAP atual 150 é "District of Columbia", enquanto o valor na versão de GAP subsequente 152 é "Florida", conforme indicado na linha 23. Em uma implementação, a entrada de diferença de GUI 3604 inclui um elemento ParentChildIdentifier na linha 3, que identifica a relação entre dois objetos de GUI em uma dada versão de GAP, de modo que restrições de classe e herança de GUI possam ser validadas (discutido em maiores detalhes abaixo).

[00275] Com referência brevemente à Figura 45, a entrada de diferença de GUI 4504 indica na linha 1 que a janela StateList na versão de GAP atual 150 corresponde à janela School na versão de GAP subsequente 152 indicada na linha 22 pelo valor de Version igual a 0 e 1, respectivamente. Os objetos de GUI StateList e School são da classe WindowsForm10.Window.8, conforme mostrado nas linhas 8 e 28. O identificador de subclasse para o objeto de GUI de StateList e School distingue os objetos de GUI (Por exemplo, app4 e app.0.0378734a, respectivamente). A entrada de diferença de GUI 4504 indica que os elementos Location das janelas StateList e School são diferentes, conforme mostrado nas linhas 7 e 27, respectivamente. Contudo, a entrada de diferença de GUI 4504 também indica que os elementos Style e ExStyle são idênticos, conforme mostrado nas linhas 9-10 e 29-30, respectivamente.

[00276] Com referência brevemente à Figura 46, a entrada de diferença de GUI 4604 indica na linha 3 que a caixa de lista SchoolListbox na versão de GAP atual 150 corresponde à caixa de

combinação SchoolCombobox na versão de GAP subsequente 152 indicada na linha 13. A entrada de diferença de GUI 4604 indica que o elemento Location da SchoolListbox e o da SchoolCombobox são diferentes, conforme mostrado nas linhas 7 e 18, respectivamente. A entrada de diferença de GUI 4604 também indica que os elementos Class, Style e ExStyle são diferentes, conforme mostrado nas linhas 8-10 e 19-21, respectivamente. Em particular, uma ou mais das propriedades de uma WindowsForms10.LISTBOX e uma WindowsForms10.COMBOBOX são diferentes, incompatíveis com as propriedades da outra classe, e os elementos de GUI filhos de objetos de GUI destas duas classes podem ter uma ou mais propriedades incompatíveis.

[00277] A Figura 37 mostra um script de teste atual 164 para o GAP atual. O script de teste atual 164 inclui declarações de navegação (por exemplo, L1 e L6) que navegam para objetos de GUI, realizam ações (funções) de leitura, escrita ou outras em objetos de GUI, e os argumentos destas funções. Por exemplo, a linha 1 do script de teste atual 164 navega para uma janela StateList, localiza 'Open File' identificado como um objeto de GUI filho da janela StateList, e realiza uma ação 'Click' no objeto de GUI 'Open File' em coordenadas XY 86, 12. Através de uma série de declarações de navegação e de ação, o script de teste atual 164 abre um arquivo 'university.data', conforme indicado pelas linhas 2-3. O script de teste atual 164 seleciona uma escola 'Acme State University' como resultado das declarações de navegação e de ação nas linhas 4-7, com base nas coordenadas 67, 12 no WinObject "Select School". O script de teste atual 164 muda a escala acadêmica para '3' como resultado das declarações nas linhas 8-10, e salva a mudança em um novo arquivo 'university_revise.data' como resultado das declarações nas linhas 11-14.

[00278] A Figura 38 ilustra uma representação de script de teste

atual 3800 que o analisador gramatical de script 166 produz como uma representação intermediária do script de teste atual 164. Em uma implementação, a SAA 108 implementa a representação de script de teste atual como uma árvore de sintaxe abstrata (AST) 168. A representação de script de teste atual 3800 representa um vetor (por exemplo, o script de teste atual 164), cujos elementos são vetores que representam declarações de navegação do script de teste atual 164. Em outras palavras, a representação de script de teste atual 3800 representa as declarações de navegação como vetores de declaração de script de teste que navegam para objetos de GUI e realizam ações naqueles objetos de GUI. A Tabela 10 ilustra uma gramática que o analisador gramatical de script 166 pode usar para produzir a representação de script de teste atual 3800. Os radicais func, action e var representam os nomes de funções específicas de plataforma que navegam para objetos de GUI (por exemplo, Window, VbEdit, e WinObject), realizam ações (funções) de leitura, escrita ou outras em objetos de GUI, e os argumentos destas funções, respectivamente.

Tabela 10 – Gramática de Analisador Gramatical de Script
navstmt::= func(arg) | navstmt. Navstmt | navstmt action arg fullnavstmt::= var = navstmt | navstmt action arg arg::= expr | "," arg |

[00279] O analisador gramatical de script 166 representa os vetores de declaração de script de teste como uma sequência ordenada de nós que contêm nomes de função e os argumentos daquelas funções que navegam para objetos de GUI. Os nós de um vetor de declaração de script de teste incluem um nó de origem e um nó de destino. Por exemplo, o analisador gramatical de script 166 pode representar o vetor de declaração de script de teste correspondente à linha 1 do script de teste atual 164 como uma StateList de nó de origem 3802 e um 'Open File' de nó de destino 3804. Os nós de um vetor de declaração de script de teste também incluem nós intermediários

posicionados entre um nó de origem e um nó de destino. Por exemplo, o analisador gramatical de script 166 pode representar o vetor de declaração de script de teste correspondente à linha 13 do script de teste atual 164 como uma StateList de nó de origem 3802, um 'Save a Data Record' de nó intermediário 3806 e 'File name' de nó de destino 3808. Em uma implementação, o analisador de script 170 usa os vetores de declaração de script de teste para analisar caminhos de navegação plausíveis para objetos de GUI identificados no modelo de diferença de GUI 162 por entradas de diferença de elemento de GUI, descritas em maiores detalhes abaixo. O analisador de script 170 pode usar os vetores de declaração de script de teste para também analisar objetos de GUI plausíveis identificados no depósito de objeto 174, também discutido em maiores detalhes abaixo.

[00280] O analisador gramatical de script 166 avalia argumentos de funções de navegação e de ação como expressões, variáveis e constantes. Os argumentos expressam as propriedades físicas de objetos de GUI para os quais os vetores de declaração de script de teste navegam e valores usados para a realização de ações naqueles objetos de GUI. Por exemplo, as coordenadas '86, 12' 3812 identificam a localização para um dispositivo de apontar para realizar uma ação 'Click' 3810 no objeto de GUI 'Open File' 3804, o qual é um objeto de GUI filho da janela StateList 3802. O analisador de script 170 usa os nomes dos objetos de GUI (por exemplo, StateList 3802 e 'Open File' 3804) navegados para pelos vetores de declaração de script de teste para localização das propriedades físicas correspondente dos objetos de GUI armazenados em um depósito de objeto (OR) 174.

[00281] Em uma implementação, o analisador de script 170 usa a lógica de OR Lookup 172 para retornar a partir do depósito de objeto 174 as propriedades físicas dos objetos de GUI navegados para por um vetor de declaração de script de teste. Em uma implementação, a

lógica de OR Lookup 172 é dividida em duas subfunções: 1) uma lógica de consulta adaptada para localizar e recuperar as propriedades físicas dos objetos de GUI navegados pelo vetor de declaração de script de teste (por exemplo, 3802-3804, 3802-3806-3808, e 3802-3814); e 2) uma lógica de localizador que encontra e retorna uma entrada de diferença de elemento de GUI (nó) no modelo de diferença de GUI 162 que corresponde ao objeto de GUI com as dadas propriedades físicas. A lógica de OR Lookup 172 pode incluir uma lógica de travessia de caminho, discutida em maiores detalhes abaixo, para a identificação de possíveis caminhos de navegação do vetor de declaração de script de teste entre um objeto de GUI de nó de origem e um objeto de GUI de nó de destino para o qual um vetor de declaração de script de teste navega.

[00282] A Tabela 11 ilustra uma implementação de um depósito de objeto 174, na forma de um esquema de XML. O depósito de objeto 174 inclui uma entrada de objeto de GUI para cada objeto de GUI da versão de GAP atual 150 identificada no script de teste atual 164. O depósito de objeto 174 pode ser gerado por uma ferramenta de escrita de script, tais como Quick Test Pro (QTP), Rational Robot, e Compuware Test Partner. O analisador de script 170 pode consultar o depósito de objeto 174 para identificar as propriedades físicas dos objetos de GUI navegados para pelos vetores de declaração de script de teste representados pela representação de script de teste atual 3800 que o analisador gramatical de script 166 produz como uma. As propriedades físicas de um objeto de GUI podem indicar se o objeto de GUI está oculto, é apenas de leitura, um número e valores-padrão, conforme mostrado na Tabela 12.

[00283] Por exemplo, o analisador de script 170 analisa os objetos de GUI 3802-3814 no vetor de declaração de script de teste. A coordenada '19,22' 3818 identifica a localização para um dispositivo de

apontar para realizar uma ação de 'Click' 3812 na SchooListbox de objeto de GUI 3814, o que é um objeto de GUI filho da janela StateList 3802. O analisador de script 170 invoca a lógica de OR Lookup 172 para localizar as propriedades físicas dos objetos de GUI 3802 e 3814. A lógica de OR Lookup 172 localiza as propriedades físicas da janela StateList 3802 e o WinObject SchoolListbox 3814, conforme mostrado na Tabela 11 nas linhas 3 e 12. O analisador de script 170 usa as propriedades físicas recuperadas a partir do depósito de objeto 174 para a localização das entradas de diferença de GUI correspondentes (por exemplo, 4504 e 4604) no modelo de diferença de GUI 162. As entradas de diferença de GUI 4504 e 4604 indicam que a janela StateList 3802 e o WinObject SchoolListbox 3814 na versão de GAP atual 150 correspondem à janela School 3402 e ao WinObject SchoolCombobox 3406 na versão de GAP subsequente 152, respectivamente. Em uma IP, o analisador de script 170 emprega a lógica de OR Lookup 172 para atravessar o modelo de diferença de GUI 162 usando as propriedades físicas dos objetos de GUI navegados pelo vetor de declaração de script de teste. A função de lógica de OR Lookup 172 retorna uma entrada de diferença de elemento de GUI (por exemplo, 3604, 4504 e 4604) a partir do modelo de diferença de GUI 162 que representa o objeto de GUI navegado pelo vetor de declaração de script de teste (por exemplo, 3802-3804-3810-3812, 3802-3806-3808-3820-3822, e 3802-3814-3826-3818).

Tabela 11 – Depósito de Objeto

```
- <XYZRep:ObjectRepository
 xmlns:XYZRep="http://www.vendorXYZ.com/XYZ/ObjectRepository">
- <XYZRep:Objects>

L3 - <XYZRep:Object Class="Window" Name="StateList">

+ <XYZRep:Properties>
+ <XYZRep:BasicIdentification>
+ <XYZRep:CustomReplay>

L7 - <XYZRep:ChildObjects>
 + <XYZRep:Object Class="WinObject" Name="Open File">
 + <XYZRep:Object Class="WinObject" Name="StateListbox">
 + <XYZRep:Object Class="WinObject" Name="Select State">
 + <XYZRep:Object Class="WinObject" Name="Select School">
 + <XYZRep:Object Class="WinObject" Name="SchoolListbox">
 </XYZRep:ChildObjects>

 </XYZRep:Object>
 </XYZRep:Objects>
 <XYZRep:Parameters />
 <XYZRep:Metadata />
 </XYZRep:ObjectRepository>
```

[00284] A Tabela 12 ilustra as propriedades físicas que podem estar localizadas no depósito de objeto para a entrada de objeto de GUI correspondente à SchoolListbox 3814.

Tabela 12 – Entrada de objeto de GUI WinObject ("SchoolListbox")

```
- <XYZRep:Object Class="WinObject" Name="SchoolListbox">
```

L2 - <XYZRep:**Properties**>

- <XYZRep:Property Name="y" Hidden="0" ReadOnly="0" Type="NUMBER">
<XYZRep:Value RegularExpression="0">86</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="x" Hidden="0" ReadOnly="0" Type="NUMBER">
<XYZRep:Value RegularExpression="0">420</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="windowstyle" Hidden="0" ReadOnly="0" Type="NUMBER">
<XYZRep:Value RegularExpression="0">1442906305</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="windowextendedstyle" Hidden="0" ReadOnly="0" Type="NUMBER">
<XYZRep:Value RegularExpression="0">512</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="window id" Hidden="0" ReadOnly="0" Type="NUMBER">
<XYZRep:Value RegularExpression="0">1182924</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="width" Hidden="0" ReadOnly="0" Type="NUMBER">
<XYZRep:Value RegularExpression="0">336</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="visible" Hidden="0" ReadOnly="0" Type="BOOL">
<XYZRep:Value RegularExpression="0">-1</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="regexpwndclass" Hidden="0" ReadOnly="0" Type="STRING">
<XYZRep:Value RegularExpression="0">WindowsForms10.LISTBOX.app4</XYZRep:Value>
</XYZRep:Property>

- <XYZRep:Property Name="object class" Hidden="0" ReadOnly="0" Type="STRING">
<XYZRep:Value RegularExpression="0">WindowsForms10.LISTBOX.app4</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="nativeclass" Hidden="0" ReadOnly="0" Type="STRING">
<XYZRep:Value RegularExpression="0">WindowsForms10.LISTBOX.app4</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="height" Hidden="0" ReadOnly="0" Type="NUMBER">
<XYZRep:Value RegularExpression="0">260</XYZRep:Value>
</XYZRep:Property>
- <XYZRep:Property Name="enabled" Hidden="0" ReadOnly="0" Type="BOOL">
<XYZRep:Value RegularExpression="0">-1</XYZRep:Value>
</XYZRep:Property>

L39 </XYZRep:**Properties**>

- <XYZRep:BasicIdentification>

<XYZRep:PropertyRef>y</XYZRep:PropertyRef>
<XYZRep:PropertyRef>x</XYZRep:PropertyRef>
<XYZRep:PropertyRef>windowstyle</XYZRep:PropertyRef>
<XYZRep:PropertyRef>windowextendedstyle</XYZRep:PropertyRef>
<XYZRep:PropertyRef>width</XYZRep:PropertyRef>
<XYZRep:PropertyRef>visible</XYZRep:PropertyRef>
<XYZRep:PropertyRef>regexpwndclass</XYZRep:PropertyRef>
<XYZRep:PropertyRef>object class</XYZRep:PropertyRef>
<XYZRep:PropertyRef>nativeclass</XYZRep:PropertyRef>
<XYZRep:PropertyRef>height</XYZRep:PropertyRef>

```
<XYZRep:PropertyRef>enabled</XYZRep:PropertyRef>

</XYZRep:BasicIdentification>

- <XYZRep:CustomReplay>
<XYZRep:Behavior Name="simclass"
Type="STRING">WindowsForms10.LISTBOX.app4</XYZRep:Behavior>
</XYZRep:CustomReplay>

- <XYZRep:Comments>
<XYZRep:Comment Name="miccommentproperty" />
</XYZRep:Comments>
<XYZRep:ChildObjects />
</XYZRep:Object>
```

[00285] A Figura 39 mostra um exemplo de um sistema analisador de script (SAS) 3900 que pode implementar o analisador de script 170. O SAS 3900 inclui uma memória 3902 acoplada a um processador 3904, e uma interface 190. Em uma implementação, a interface 190 se comunica com o depósito de metadados de elemento de GUI 138 e um comparador de GUI 160 para recebimento de metadados de elemento de GUI 140 e do modelo de diferença de GUI 162, respectivamente. A interface 190 é conectada a uma rede 3906 (por exemplo, a Internet) em comunicação com vários outros sistemas e recursos. Em uma outra implementação, a memória 3902 inclui os metadados de elemento de GUI 140 e uma lógica de modelo de diferença de GUI 3908 que produz o modelo de diferença de GUI 162 e entradas de diferença de elemento de GUI 3910. A memória 3902 também inclui uma lógica de analisador gramatical de script 3912 que recebe o script de teste atual 164 e produz a AST 168, processa a AST 168 como uma representação de script de teste atual 3914, e

produz os vetores de declaração de script de teste 3916 (por exemplo, 3802-3804-3810-3812, 3802-3806-3808-3820-3822, e 3802-3814-3826-3818).

[00286] A memória 3902 ainda inclui uma lógica de análise de script 3920 que recebe o modelo de diferença de GUI 162, e a representação de script de teste atual 3914 para o script de teste atual 164, incluindo um vetor de declaração de script de teste 3916. Em uma implementação, a lógica de análise de script 3920 invoca a lógica de OR Lookup 172 para localizar, no depósito de objeto 174, uma entrada de objeto de GUI 3922 combinando com o vetor de declaração de script de teste 3916. Em uma implementação, o analisador de script 170 invoca a lógica de OR Lookup 172 para localizar, em várias fontes externas, uma entrada de objeto de GUI 3922 combinando com o vetor de declaração de script de teste 3916. Quando o vetor de declaração de script de teste 3916 (por exemplo, 3802-3804-3810-3812, 3802-3806-3808-3820-3822, e 3802-3814-3826-3818) emprega constantes para a identificação de nomes de objeto de GUI, ao invés de expressões cujos valores podem ser determinados apenas no tempo de execução, a função de OR Lookup 172 pode usar o nome de objeto de GUI e as propriedades do objeto de GUI para localizar de forma eficiente a entrada de objeto de GUI correta 3922 e localizar, no modelo de diferença de GUI 162, uma entrada de diferença de elemento de GUI 3910 combinando com a entrada de objeto de GUI 3922.

[00287] Por exemplo, o vetor de declaração de script de teste representado por 3802-3804-3810-3812 identifica a janela StateList de objeto de GUI 3302 e o objeto de GUI de caixa de lista SchoolListbox 3316, mostrados na declaração de navegação de script de teste atual 164 mostrada na linha 6 da Figura 37:

[00288] Window("StateList").WinObject("SchoolListbox").Click

19,22.

[00289] A função de OR Lookup 172 localiza a entrada de objeto de GUI 3922 para cada objeto de GUI 3302 e 3316, usando os nomes conhecidos dos objetos de GUI, StateList e SchoolListbox, respectivamente. A função de OR Lookup 172 localiza as entradas de diferença de elemento de GUI correspondentes 4504 e 4604, no modelo de diferença de GUI 162. A lógica de análise de script 3920 extrai uma declaração de script de teste transformado 3926 que corresponde a 3402 e 3406:

[00290] Window("School").WinObject("SchoolCombobox").Click 294,14.

[00291] A lógica de regra de classe de GUI 3928 pode usar os nomes de objeto de GUI para localização das propriedades usadas para se validar que a declaração de script de teste transformado 3928 não viola as regras de mudança de script de elemento de GUI 194 e as restrições. Em uma implementação, o analisador de script 170 usa a lógica de regra de classe de GUI 3928 em conjunto com o motor de satisfação de restrição 188 para determinação de violações das regras de mudança de script de elemento de GUI 194 e restrições.

[00292] Por exemplo, quando a declaração de script de teste transformado 3926 acessa objetos de GUI que não existem na versão de GAP subsequente 152 e/ou a declaração de script de teste transformado 3926 tenta regular um valor de um objeto de GUI que não é compatível com a classe de GUI daquele objeto de GUI, então, a declaração de script de teste transformado 3926 viola as restrições impostas ao objeto de GUI. O motor de satisfação de restrição 188 valida a declaração de script de teste transformado 3926 para ajudar a verificar que operações incorretas sejam identificadas na declaração de script de teste transformado 3926 para um programador resolver. Em uma implementação, o motor de satisfação de restrição 188

recebe uma mensagem de inquisição de compatibilidade (por exemplo, a partir de um sistema externo, tal como o depósito de metadados de elemento de GUI 138) que especifica dois tipos de elemento de GUI. O CSE 188 analisa os dois tipos e retorna uma mensagem de verificação de compatibilidade que indica se os dois tipos são compatíveis. No caso de a mudança de objeto de GUI violar uma regra de satisfação de restrição 3932, então, a mensagem de verificação de compatibilidade proverá uma informação detalhada referente à violação.

[00293] O motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 podem inferir uma informação de classe de GUI referente a objetos de GUI que estejam presentes em um caminho de navegação de vetores de declaração de script de teste 3916 e declarações de script de teste transformado 3926, e cuja presença não é explicitamente definida. Em uma implementação, o motor de satisfação de restrição 188 e/ou a lógica de regra de classe de GUI 3928 usam uma combinação de validação de tipo de tempo de compilação e uma validação de inferência de classe de GUI, de modo a validarem a correção de vetores de declaração de script de teste 3916 e declarações de script de teste transformado 3926. Por exemplo, quando o vetor de declaração de script de teste 3916 emprega expressões que identificam as possíveis entradas de objeto de GUI correspondentes 3922 e as entradas de diferença de elemento de GUI 3910 no depósito de objeto 174 e no modelo de diferença de GUI 162, respectivamente. A lógica de regra de classe de GUI 3928 e o motor de satisfação de restrição 188 então identificam as entradas de objeto de GUI válidas 3922 que podem substituir as expressões e as entradas de diferença de elemento de GUI 3910 que satisfazem aos vetores de declaração de script de teste 3916 e às declarações de script de teste transformado 3926. De modo similar, quando a

declaração de script de teste transformado 3928 emprega expressões que identificam objetos de GUI cujos valores podem ser determinados apenas no tempo de execução, a função de OR Lookup 172 usa a lógica de travessia de caminho 3924 para identificar todas as entradas de diferença de elemento de GUI correspondentes 3910 que identifiquem objetos de GUI que possam substituir as expressões, e a lógica de regra de classe de GUI 3928 e o motor de satisfação de restrição 188 validam cada substituição de objeto de GUI pelas expressões usadas para a formação da declaração de script de teste transformado 3928.

[00294] Por exemplo, considere a declaração de script de teste transformado 3928: VBWindow("s").VBWindow(e1).VBWindow(e2).VBWindow("d"), onde o objeto de GUI de nó de origem é denominado "s", o objeto de GUI de nó de destino é denominado "d", mas as expressões e1 e e2 computam valores de nós intermediários no caminho de navegação no tempo de execução. A lógica de travessia 3924 determina nós intermediários (objetos de GUI) que podem ser incluídos nos caminhos de navegação identificados pelo nó de origem "s" e pelo nó de destino "d". A lógica de travessia de caminho 3924 analisa o modelo de diferença de GUI 162 para identificar possíveis substituições de constante por e1 e e2, por exemplo, "a" e "f", de modo que a declaração de script de teste transformado 3928 formada pelos objetos de GUI substitutos na expressão de caminho de navegação "s.a.f.d" possa ser validada pela lógica de regra de classe de GUI 3928 e/ou pelo motor de satisfação de restrição 188. Pela identificação dos possíveis caminhos de navegação levando ao nó de destino d começando com o nó de origem s, a lógica de regra de classe de GUI 3928 e o motor de satisfação de restrição 188 podem concluir se a declaração de script de teste transformado 3928 é válida. No caso de

a lógica de travessia 3924 não identificar pelo menos um caminho de navegação, então, a declaração de script de teste transformado 3928 será inválida. Alternativamente, no caso de a lógica de travessia 3924 identificar caminhos de navegação levando a partir de s para d pela travessia de dois objetos (por exemplo, e1 e e2), então, a declaração de script de teste transformado 3928 poderá ser válida, desde que as expressões e1 e e2 avaliem para os nomes dos nós nos caminhos de navegação descobertos. A lógica de travessia 3924 infere os nomes possíveis computados pelas expressões e1 e e2 no tempo de compilação.

[00295] Uma descrição formal da lógica de travessia 3924 é provida com referência à Figura 47. As expressões e1 e e2 podem ser substituídas pelas variáveis de nome de objeto α e β de forma correspondente, e a expressão original é convertida na estratégia de travessia S = s → α → β → d. A função 'first(s)' computa um conjunto de bordas que podem ser atravessadas a partir do nó s. Estas bordas levam a um conjunto de objetos designados pela variável α. A função 'first(s)' pode ser computada usando-se um algoritmo de capacidade de alcance de gráfico, incluído na lógica de travessia de caminho 3924, e a lógica de travessia de caminho 3924 retorna bordas que para que se pode navegar para o nó de destino. De acordo com a Figura 47, α = {a, b, c}. Então, para cada elemento de α, função 'first' pode ser computada. Como resultado, β = {e, f, g} são obtidos, onde 'first(a)' = {e, g}, 'first(b)' = {e}, e 'first(c)' = {f}, e 'first(e)' = {Ø}, 'first(f)' = {d}, e 'first(g)' = {d}. A partir dos valores de nó computados a lógica de travessia de caminho 3924 forma uma lista de trabalho W que inclui um conjunto de todos os caminhos computados, W= {(s, a, e), (s, a, g, d), (s, b, e), (s, c, f, d)}. A lógica de travessia de caminho 3924 analisa cada caminho de navegação de W para determinar se o caminho de

navegação contém os nós s e d. Os caminhos de navegação identificados pela lógica de travessia de caminho 3924 como incluindo os nós s e d, como nós de origem e de destino, são considerados como caminhos de navegação válidos. No caso de nenhum caminho de navegação ser identificado pela lógica de travessia de caminho 3924, então, a declaração de script de teste transformado 3928 será inválida, porque o elemento de GUI alvo não pode ser alcançado começando-se a partir do elemento de GUI de começo especificado. A lógica de travessia 3924 de modo similar valida os vetores de declaração de script de teste 3916.

[00296] Com referência de novo à Figura 47, um exemplo de uma expressão inválida é VBWindow("s").VBWindow(e1).

[00297] VBWindow(e2).VBWindow(e3).VBWindow("d"). Todos os caminhos de navegação entre os nós s e d têm no máximo dois objetos. Portanto, não importando que valores são computados no tempo de execução para as expressões e1, e2 e e3, as expressões não podem representar objetos em um caminho de navegação válido entre os objetos de origem e de destino. U outro exemplo de uma expressão inválida é

[00298] VBWindow("s").VBWindow("b").VBWindow(e1).VBWindow("d"),

[00299] porque nenhum valor para a expressão e1 existe que torne o caminho de navegação válido (isto é, que forme um caminho completo de 's' para 'd').

[00300] O motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 podem inferir uma informação de classe de GUI referente a objetos de GUI que estejam presentes no caminho de navegação de vetores de declaração de script de teste 3916 e declarações de script de teste transformado 3926. Um recurso da SAA 108 é manter a compatibilidade de operações nos objetos de GUI

entre scripts de teste sucessivos. Quando uma declaração de script de teste transformado 3926 tenta acessar objetos de GUI que não existem em uma versão GAP subsequente 152 e/ou tenta regular um valor de um objeto de GUI que não é compatível com o tipo do objeto de GUI, a declaração de script de teste transformado 3926 viola as entradas de regras de mudança de script de elemento de GUI 3930 e/ou as regras de satisfação de restrição 3932 impostas no objeto de GUI. O motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 podem checar cada declaração de script de teste transformado potencial 3926, antes da inclusão do vetor no script de teste transformado 178, de modo que operações inválidas possam ser identificadas e mensagens de guia de mudança correspondentes 3938 possam ser extraídas.

[00301] O motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 usam relações de herança e subtipificação entre as classes de objetos de GUI. O conceito de classe inclui contenções hierárquicas (por exemplo, escopos de GUI e hierarquias de sistema). O depósito de objeto 174 e o modelo de diferença de GUI 162 incluem uma informação de classe de GUI (por exemplo, anotação das classes de objetos de GUI) para cada entrada de objeto de GUI 3922 e entrada de diferença de elemento de GUI 3910. Por exemplo, com referência à linha 1 da Tabela 12, a SchoolListBox é uma classe de WinObject com propriedades listadas nas linhas 3-39. Em um outro exemplo, com referência às Figuras 36, 45 e 46, na linha 1 de cada entrada de diferença de GUI (por exemplo, 3604, 4504 e 4604) o GUIElement Type é indicado. A classe de cada objeto de GUI é indicada conforme mostrado nas Figuras 36, 45 e 46, nas linhas 7, 8 e 8, respectivamente. A classe de um objeto de GUI indica que o objeto de GUI inclui atributos, propriedades e/ou tratos em particular em comum com outros objetos de GUI da mesma classe que podem ser

estendidos e/ou herdados pelos objetos de GUI filhos. Por exemplo, a Figura 36 na linha 7 indica que o objeto de GUI StateListbox é de uma classe WindowsForms10.ListBox.app4 que inclui valores, conforme indicado na linha 11 da Figura 36. Em outras palavras, uma propriedade de objetos de GUI de WindowsForms10.ListBox.app4 é que estes objetos de GUI são esperados como tendo valores. A classe é um conceito que uma estrutura de GUI usa para a classificação de objetos de GUI. Por exemplo, a classe ListBox define formato, funcionalidade e as regras de interatividade para objetos de GUI nesta classe. A atribuição de classes a objetos de GUI facilita o motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 a rastrearem mudanças entre versões de GAP sucessivas (por exemplo, 150 e 152) e a realizarem uma checagem estendida na correção de operações em objetos de GUI.

[00302] Com referência de novo à Figura 39, em uma implementação, a lógica de regra de classe de GUI 3928 e o motor de satisfação de restrição 188 determinam se um objeto de GUI mudou e regulam o status de mudança de elemento de GUI 3934. Por exemplo, o status de mudança de elemento de GUI 3934 pode usar um indicador numérico de 0, 1 e 2, respectivamente, para indicar que um objeto de GUI não foi mudado, mudou com ou mudou sem violações de regras de mudança de script de elemento de GUI 194 e/ou regras de satisfação de restrição 3932. A lógica de análise de script 3920 pode usar o status de mudança de elemento de GUI 3934, as entradas de regras de mudança de script de elemento de GUI 3930 e as regras de satisfação de restrição 3932 para buscar no depósito de mensagem de guia de mudança 192 e identificar o especificador de mudança apropriado 184 e os identificadores de mensagem de guia de mudança 3936. As entradas de regras de mudança de script de elemento de GUI 3930 podem indicar se um elemento de um objeto de GUI pode

ser mudado de uma forma em particular. Por exemplo, as regras de mudança de script de elemento de GUI 3930 podem indicar que um elemento de um objeto de GUI em particular não pode ser mudado apenas de leitura para editável. Em um outro exemplo, uma mudança para a classe de um objeto de GUI pode resultar em um objeto de GUI filho violando uma regra de mudança de script de elemento de GUI 3920, porque um ou mais atributos do objeto de GUI filho estão em conflito com uma mudança de classe do elemento de GUI pai. Além disso, um botão pode ser substituído por itens de menu, e ações que são corretas para acesso e manipulação de botões não funcionarão para menus.

[00303] Em uma outra implementação, o status de mudança de elemento de GUI 3934 é uma mensagem que provê uma descrição detalhada da mudança. O status de mudança de elemento de GUI 3934 também pode indicar com um indicador numérico (por exemplo, -1) que o objeto de GUI foi apagado da versão de GAP subsequente 152. Quando um objeto de GUI foi apagado da versão de GAP subsequente 152 e uma declaração de script de teste transformado 3926 inclui uma referência ao objeto de GUI, a lógica de análise de script 3920 extrai uma mensagem de guia de mudança 3938 que indica que o objeto de GUI foi apagado.

[00304] Em uma implementação, a lógica de análise de script 3920 extrai um especificador de mudança 184 que inclui uma declaração de script de teste transformado 3926 que viola uma regra de mudança de script de elemento de GUI 194 e/ou uma regra de satisfação de restrição 3932, de modo que um programador pode avaliar se é para modificar a declaração de script de teste transformado 3926 e/ou a versão de GAP subsequente 152 para a obtenção de um resultado desejado e/ou se adequar a uma exigência desejada que, de outra forma, pode não ter sido evidente. Em uma outra implementação, a

lógica de análise de script 3920 proíbe a saída de declarações de script de teste transformado 3926 que violem certas regras de mudança de script de elemento de GUI 194 e regras de satisfação de restrição 3932. Cada uma das regras de mudança de script de elemento de GUI 194 e regras de satisfação de restrição 3932 pode incluir indicadores que indiquem o nível ou a severidade de uma violação e se a lógica de análise de script 3920 pode extrair uma declaração de script de teste transformado 3926, embora uma violação tenha ocorrido. Independentemente de a lógica de análise de script 3920 extrair uma declaração de script de teste transformado 3926, a lógica de análise de script 3920 pode extrair um guia de mudança 180 que inclua mensagens de guia de mudança 3938 correspondentes a cada uma das violações.

[00305] Para cada status de mudança de elemento de GUI 3934 que indique uma mudança na violação de regras de mudança de script de elemento de GUI 194 e/ou regras de satisfação de restrição 3932, a lógica de análise de script 3920 extrai um conjunto de identificadores de mensagem de guia de mudança 3936 e mensagens de guia de mudança correspondentes 3938. As mensagens de guia de mudança 3938 podem incluir o tipo de apagamento de caminho errado (WP-1) 3940, o tipo de mesmo caminho errado (WP-2) 3942 e o tipo de elemento mudado (CE-1 e CE-2) 3944. O guia de mudança 180 e as mensagens de guia de mudança 3938, incluindo 3940, 3942 e 3944, são descritas em mais detalhes abaixo.

[00306] A Figura 40 mostra um fluxograma 900 para a recuperação das propriedades de uma entrada de objeto de GUI 3922 a partir de um depósito de objeto (OR) 174. A lógica de analisador gramatical de script 3912 analisa gramaticalmente o vetor de declaração de script de teste 3916 em uma sequência ordenada de nós que representam funções e argumentos que navegam para objetos de GUI (4002). A

lógica de analisador gramatical de script 3912 avalia o primeiro nó da sequência ordenada de nós (4004) e identifica o primeiro nó como o nó de origem e atribui um identificador de sequência indicando a posição do nó de origem na sequência ordenada (4006). A lógica de analisador gramatical de script 3912 avalia o próximo nó da sequência ordenada de nós para determinar se o próximo nó é o último nó (4008), e identifica o próximo nó como um nó intermediário, quando o próximo nó não for o último nó (4010). Ao nó intermediário é atribuído um identificador de sequência indicando a posição do nó intermediário na sequência ordenada. A lógica de analisador gramatical de script 3912 pode identificar todos os nós intermediários entre o nó de origem e o nó de destino.

[00307] A lógica de analisador gramatical de script 3912 identifica o último nó na sequência ordenada como o nó de destino e atribui um identificador de sequência ao nó de destino que indica a posição do nó de destino na sequência ordenada (4012). A OR Lookup 172 realiza uma consulta de depósito de objeto para cada objeto de GUI correspondente à sequência ordenada de nós para os quais o vetor de declaração de script de teste navega, de modo que cada entrada de objeto de GUI 3922 seja identificada (4014). Em uma implementação, a sequência ordenada de nós é usada pela lógica de travessia de caminho 3924, pela lógica de regras de classe de GUI 3928 e/ou pelo motor de satisfação de restrição 188 para validação das declarações do script de teste atual 164. Em uma implementação, o analisador de script 170 usa a sequência ordenada de nós para inferir a classe de GUI e uma informação de herança (subclasse) para objetos de GUI. Quando pelo menos um dentre os nós de origem, de destino e/ou intermediário é de expressões que podem ser prontamente identificadas no tempo de execução, a lógica de travessia de caminho pode identificar possíveis entradas de objeto de GUI 3922, e a lógica

de regra de classe de GUI 3928 e/ou o motor de satisfação de restrição 188 determinam as entradas de objeto de GUI 3922 que satisfazem o vetor de declaração de script de teste 3916, sem violação da lógica de regra de classe de GUI 3928 e das regras de satisfação de restrição 3932. A OR Lookup 172 recupera as propriedades das entradas de objeto de GUI 3922 para as quais o vetor de declaração de script de teste navega (4016).

[00308] A Figura 41 mostra um fluxograma 4100 para a identificação de uma entrada de diferença de elemento de GUI 3910 correspondente a uma entrada de objeto de GUI 3922. A OR Lookup 172 recebe as propriedades dos objetos de GUI correspondentes aos nós de origem, de destino e todos os intermediários do vetor de declaração de script de teste 3916. Em uma implementação, a OR Lookup 172 emprega a lógica de travessia de caminho 3924 para a identificação de possíveis entradas de diferença de elemento de GUI 3910 correspondentes aos caminhos de navegação identificados por um nó de origem e um nó de destino para o qual um vetor de declaração de script de teste navega (4102). Quando pelo menos uma das entradas de diferença de elemento de GUI 3910 é uma expressão que pode ser identificada apenas no tempo de execução, a lógica de travessia de caminho 3924 identifica uma ou mais entradas de diferença de elemento de GUI 3910 que formam um caminho de navegação entre o nó de origem e o nó de destino (4104). A lógica de travessia de caminho 3924 determina se as entradas de diferença de elemento de GUI 3910 formam um caminho de navegação válido entre as entradas de diferença de elemento de GUI 3910 de nós de origem e de destino correspondentes (4106). A lógica de regra de classe de GUI 3928 e/ou o motor de satisfação de restrição 188 determinam se as entradas de diferença de elemento de GUI 3910 que formam o caminho de navegação violam a lógica de regra de classe de GUI

3928 (4108) e a regras de satisfação de restrição 3932 (4110).

[00309] A lógica de regra de classe de GUI 3928 e/ou o motor de satisfação de restrição 188 indicam as entradas de diferença de elemento de GUI 3910 que correspondem a cada uma das entradas de objeto de GUI 3922 formando um caminho de navegação válido (4112). A lógica de análise de script 3920 determina o especificador de mudança de saída com base no tipo de mudança de elemento de GUI (por exemplo, o status de mudança de elemento de GUI 3934) indicado pelos resultados da lógica de regras de classe de GUI 3928 e do motor de satisfação de restrição 188, com base na entrada de diferença de elemento de GUI 3910 (4114).

[00310] Quando a lógica de travessia de caminho 3924 identifica um caminho de navegação que atravessa um número inválido de entradas de diferença de elemento de GUI 3910 entre as entradas de diferença de GUI 3910 de nó de origem e de destino correspondentes, a lógica de travessia de caminho 3924 indica que o caminho de navegação é inválido (4116). Em uma implementação, os caminhos de navegação inválidos não são analisados pela lógica de regras de classe de GUI 3928 e pelo motor de satisfação de restrição 188, e uma declaração de script de teste transformado 3926 não é extraída como resultado. Em uma implementação, a lógica de travessia de caminho 3924 extrai uma mensagem de aviso identificando os caminhos de navegação inválidos. Em uma outra implementação, o guia de mudança inclui uma mensagem de aviso indicando os caminhos de navegação inválidos. O guia de mudança 180 pode incluir várias mensagens de aviso que refletem qualquer número de condições identificadas durante um processamento pelo analisador de script 170. O guia de mudança 180 pode incluir mensagens de aviso e/ou de informação correspondentes às condições encontradas pelo analisador de script 170 e/ou qualquer uma das lógicas do analisador

de script 170 (por exemplo, a lógica de analisador gramatical de script 3912, a lógica de análise de script 3920, a lógica de OR Lookup 172, a lógica de travessia de caminho 3924 e a lógica de regras de classe de GUI 3928) e/ou o motor de satisfação de restrição 188. A lógica de análise de script 3920 pode extrair um guia de mudança 180 com mensagens de guia de mudança 3938 (por exemplo, 3940, 3942 e 3944) correspondentes às violações de lógica de regras de classe de GUI 3928 e de regras de satisfação de restrição 3932 (4118).

[00311] A Figura 42 mostra um script de teste transformado 178 para a versão de GAP subsequente 152. O script de teste transformado 178 pode incluir declarações automaticamente transformadas pela lógica de análise de script 3920. O script de teste transformado 178 também pode incluir declarações introduzidas manualmente após uma revisão do guia de mudança 180. No exemplo mostrado na Figura 42, a lógica de análise de script 3920 extrai o script de teste transformado 178 com as declarações de script de teste transformado nas linhas 1, 6-11, 14-19, e 21-22, enquanto um escritor de script manualmente introduziu as linhas 2-4. A lógica de análise de script 3920 determina as declarações de script de teste transformado 3926 a extrair para o script de teste transformado 178 com base na lógica de regras de classe de GUI 3928 e nas regras de mudança de script de elemento de GUI 194. A complexidade da lógica de regras de classe de GUI 3928 e das regras de mudança de script de elemento de GUI 194 e a riqueza do modelo de diferença de GUI 162 e dos metadados de elemento de GUI podem ser especificadas em qualquer nível de complexidade para condução quanto se puder do script de teste transformado 178 a lógica de análise de script 3920 extrair sem uma entrada de programador.

[00312] Por exemplo, a lógica de análise de script 3920 transforma as linhas 1-3 do script de teste atual 164, mostrado na Figura 37, em

declarações de script de teste transformado 3926, linhas 1, 5-7, mostradas na Figura 42. um programador pode introduzir as linhas 2-4 do script de teste transformado 178, porque aquelas linhas incluem objetos de GUI novos (por exemplo, "ToolbarWindow32" e valores "My Computer") para os quais o modelo de diferença de GUI 162 pode não incluir uma informação que identifique um objeto de GUI correspondente na versão de GAP atual 150. Em uma implementação, o modelo de diferença de GUI 162 e os metadados de elemento de GUI provêm a classe de GUI, a tipificação de GUI e uma informação de mapeamento necessárias para a lógica de análise de script 3920 inferir as linhas 2-4 do script de teste transformado 178, dado que "university.data" na linha 6 representa um destino em uma travessia de caminho a partir da qual declarações de script de teste intermediárias podem ser determinadas. Em um outro exemplo, o modelo de diferença de GUI 162 e/ou os metadados de elemento de GUI incluem uma classe de GUI e uma informação de mapeamento que o analisador de script 170 usa para a transformação da linha 11 do script de teste atual 164 que refere-se ao WinObject "Save File" nas declarações de script de teste transformado 3926 nas linhas 16-17 que referem-se a um objeto de GUI filho "Save File" do WinObject "menuStrip1".

[00313] A Figura 43 mostra um guia de mudança 180 que o analisador de script 180 pode extrair. Em uma implementação, a lógica de análise de script 3920 extrai um especificador de mudança 184 que indica se a declaração de script de teste transformado 3926 viola uma regra de mudança de script de elemento de GUI 194 com base em uma análise realizada pela lógica de regras de classe de GUI 3928 e/ou pelo motor de satisfação de restrição 188. O analisador de script 170 extrai especificadores de mudança 184 que também podem incluir as declarações de script de teste transformado 3926. O analisador de

script 170 determina as modificações nos objetos de GUI e nas declarações de script de teste transformado 3928 que são afetadas pelas modificações nos objetos de GUI aos quais as declarações de script de teste transformado 3928 referem-se. Em uma implementação, a lógica de análise de script 184 determina os objetos de GUI referidos no script de teste atual 164 que são apagados na versão de GAP subsequente 152. O analisador de script 170 extrai os especificadores de mudança 184 que podem incluir as mensagens de guia de mudança 3938, as declarações de script de teste transformado 3926 e/ou ambas. Em uma implementação, o analisador de script 170 extrai especificadores de mudança 184 que incluem identificadores de mensagem de guia de mudança 3930, declarações de script de teste transformado 3926 e/ou ambas.

[00314] Alguns dos tipos de mudanças em objetos de GUI entre liberações para edição sucessivas de GAPs que o analisador de script 170 pode identificar em uma mensagem de guia de mudança 3938 incluem: (A) um novo objeto de GUI adicionado a uma versão de GAP subsequente 152; (B) um objeto de GUI é apagado de uma versão de GAP subsequente 152; (C) os valores de um ou mais atributos de um objeto de GUI são modificados; (D) os valores de um objeto de GUI são diferentes; e (E) o tipo de um objeto de GUI é diferente. As mudanças de objeto de GUI dos tipos A e B ocorrem quando objetos de GUI são adicionados e removidos de forma correspondente a partir de uma versão de GAP atual 150 e uma versão de GAP subsequente 152. Por exemplo, a adição de WinObject menustrip1 3408 à versão de GAP subsequente 152 é uma mudança de tipo A, enquanto a remoção do WinObject "Select School" 3304 é uma mudança de objeto de GUI de tipo B. Com referência à Figura 4, note que "Select School" foi removido em 412.

[00315] Um exemplo de uma mudança de tipo C é a mudança do

nome de janela de StateList 3302 para School 3402. A adição ou a remoção de valores de objetos de GUI tais como caixas de lista e de combinação é um exemplo de modificações da mudança de tipo D. Por exemplo, a caixa de lista StateListbox 3312 na versão de GAP atual 150 é identificada na versão de GAP subsequente 152 como StateListbox 3404, e com referência à entrada de diferença de GUI 3604, os valores para SeqNumber = "8" são "District of Columbia" e "Florida" para versões de GAP sucessivas, respectivamente. A mudança do "tipo" de um objeto de GUI pode incluir a substituição de uma classe da janela que é usada para representação do objeto e/ou pela mudança do conceito de nível alto que descreve os valores que o objeto de GUI assume. Por exemplo, mudar o tipo de rótulo estático para uma caixa de combinação apenas de leitura é uma modificação do tipo E. Um outro exemplo de uma mudança do tipo E inclui a mudança da caixa de lista "SchooListbox" 3316 para uma caixa de combinação "SchoolCombobox" 3406.

[00316] O analisador de script 170 classifica os tipos de mudanças que a lógica de regras de classe de GUI 3928 e/ou o motor de satisfação de restrição 188 identificam, incluindo: erros de tipo 1 de caminho errado (WP-1) que ocorrem quando um script acessa os objetos de GUI que podem estar apagados na versão de GAP subsequente 152 (por exemplo, veja "Select School" 3318 e 412 conforme mostrado na Figura 4). Os erros de WP tipo 2 (WP-2) ocorrem em scripts que são lidos ou escritos em objetos de GUI errados. Os erros de WP-2 ocorrem quando as declarações de script de teste transformado 3926 navegam para objetos de GUI errados e leem os valores do objeto de GUI errado e/ou invocam métodos no objeto de GUI errado.

[00317] Por exemplo, considere a declaração nas linhas 2 e 6 do script de teste atual 164 e do script de teste transformado 178,

respectivamente:

[00318] Window("StateList").Dialog("Open").WinListView("SysListView32").Select "university.data".

[00319] A declaração seleciona "university.data" a partir de uma WinListView "SysListView32". Contudo, as linhas 2-4 do script de teste transformado 178 podem navegar para e invocar o método Select no objeto de GUI errado "university.data", porque os objetos de GUI referenciados nas linhas 2-4 do script de teste transformado 178 são novos objetos de GUI que não são referenciados no script de teste atual 164. Assim, quando as propriedades de objetos de GUI existentes são modificadas e/ou outros objetos de GUI são adicionados em uma versão de GAP subsequente 152, o resultado de interferência destas operações é que a declaração de script de teste transformado 3926 pode acessar e ler valores de objetos que são diferentes daqueles conforme pretendido originalmente.

[00320] Erros de Elemento Mudado (CE) ocorrem quando declarações tentam acessar objetos de GUI cujos tipos, propriedades ou valores-padrão são mudados na versão de GAP subsequente 152 (CE-1). Por exemplo, a entrada de diferença de GUI 3604 indica que há valores diferentes para SeqNumber = "8" que são "District of Columbia" e "Florida" para versões de GAP sucessivas, e a lógica de análise de script 3920 pode emitir, consequentemente, uma mensagem de guia de mudança 3944 indicando que um erro de tipo CE-1 ocorreu.

[00321] A lógica de análise de script 3920 pode emitir uma mensagem de guia de mudança 3944 correspondente a um erro de tipo CE-2, quando uma declaração de script de teste transformado 3926 tentar uma operação em um objeto de GUI que não leve em consideração novas restrições impostas nos elementos, por exemplo, tentar escrever dados em uma caixa de texto apenas de leitura. Com

referência à entrada de diferença de elemento de GUI mostrada na Tabela 13, o WinObject "AcadScale" referido no script de teste atual 164 na linha 8 é um objeto editável que foi transformado no WinObject "Academics (1-5)" na versão de GAP subsequente 152, onde o objeto é apenas de leitura. Em uma implementação, a lógica de análise de script 3920 extrai uma mensagem de guia de mudança 3944 para indicar que uma mudança de tipo de CE-2 ocorreu.

Tabela 13 – Entrada de Diferença de Elemento de GUI para AcadScale

```
<GUIElement Type = "AcadScale Textbox">
<Version> 0
 - <GUIElement Alias="AcadScale">
   <UniqueID>0xcb</UniqueID>
   <HWND>0x1a0b3c</HWND>
   <Location x="573" y="790" width="32" height="23" />
   <Class>WindowsForms10.EDIT.app4</Class>
   <Style>0x560100c0</Style>
   <ExStyle>0xc0000a00</ExStyle>
 - <GUIElement Alias="AcadScale">
   <UniqueID>0x4</UniqueID>
   <HWND>0x1a0b3c</HWND>
   <Location x="575" y="792" width="28" height="19" />
   <Class>WindowsForms10.EDIT.app4</Class>
   <Style>0x560100c0</Style>
   <ExStyle>0xc0000a00</ExStyle>
 - <Values>
   <Value SeqNumber="3" />
   </Values>
   </GUIElement>
   </GUIElement>
   </Version>
```

```
<Version> 1
- <GUIElement Alias="Academics (1-5)">
  <UniqueID>0x2ff</UniqueID>
  <HWND>0x70d0e</HWND>
  <Location x="597" y="388" width="111" height="17" />
  <Class>WindowsForms10.STATIC.app.0.378734a</Class>
  <Style>0x5600000d</Style>
  <ExStyle>0xc0000800</ExStyle>
- <GUIElement Alias="Academics (1-5)">
  <UniqueID>0x308</UniqueID>
  <HWND>0x70d0e</HWND>
  <Location x="597" y="388" width="111" height="17" />
  <Class>WindowsForms10.STATIC.app.0.378734a</Class>
  <Style>0x5600000d</Style>
  <ExStyle>0xc0000800</ExStyle>
 - <Values>
  <Value SeqNumber="3">Academics (1-5)</Value>
  </Values>
  </GUIElement>
  </GUIElement>
 </Version>
</GUIElement>
```

[00322] Conhecer o tipo de modificação para um objeto de GUI facilita a lógica de análise de script 3920 na determinação da mensagem de guia de mudança apropriada 3938 e nas declarações de script de teste transformado 3926 a extrair. Por exemplo, quando uma declaração de script de teste transformado 3926 tenta regular valores em um objeto de caixa de texto, embora o tipo de objeto tenha mudado para uma caixa de combinação apenas de leitura, a lógica de análise de script 3920 extrai mensagens de guia de mudança 3944 que sugerem como modificar a declaração de script de teste

transformado 3926 para valores de seleção na caixa de combinação usando interfaces apropriadas.

[00323] A Figura 44 mostra um fluxograma para extração de declarações de script de teste transformado 3926. A lógica de análise de script 3920 revê o status de mudança de elemento de GUI 3934 de cada objeto de GUI referido no script de teste transformado 178, na versão de GAP atual 150 e/ou na declaração de script de teste transformado 3926 (4402). A lógica de análise de script 3920 determina se os objetos de GUI da declaração de script de teste transformado foram adicionados à versão de GAP subsequente 152 (4404). A lógica de regras de classe de GUI 3928 e/ou o motor de satisfação de restrição 188 podem analisar os objetos de GUI adicionados para determinarem se um erro de WP-2 ocorreu como resultado das declarações de script de teste transformado 3926 navegando para objetos de GUI errados (por exemplo, dois objetos de GUI incorretamente usando aliases idênticos) e leem os valores do objeto de GUI errado e/ou invocam métodos no objeto de GUI errado (4406). A declaração de script de teste transformado 3926 é incluída no script de teste transformado 178 e mensagens de guia de mudança 3938 correspondentes são extraídas (4408).

[00324] Quando a lógica de análise de script 3920 determina que objetos de GUI foram removidos de uma versão de GAP atual 150 (4410). A lógica de análise de script 3920 localiza as declarações que referenciam estes objetos removidos no script de teste atual 178. O analisador de script 170 refere-se a estas declarações como declarações de primeira referência (FRS) (4412). As variáveis usadas nestas declarações são obtidas, e as declarações que usam as variáveis cujos valores são definidos nas FRSs são referidas como declarações de referência secundária (SRS) (4414). A lógica de regras de classe de GUI 3928 e/ou o motor de satisfação de restrição 188

podem analisar os objetos de GUI para determinarem se erros de WP-1 ocorreram, com base nas declarações de script tentando acessar objetos de GUI que foram apagados em uma versão de GAP subsequente 152 (4416). Quando uma declaração do script de teste atual 178 refere-se a uma variável cujo valor aponta para um objeto de GUI removido, a declaração do script de teste atual 3926 é considerada uma SRS. Em uma implementação, o analisador de script 170 extrai as SRSs identificadas como declarações de script de teste transformado 3926 e mensagens de guia de mudança 3938 correspondentes, de modo que o pessoal de teste possa recuperar e decidir como modificar as SRSs (4408).

[00325] Quando os valores de um ou mais atributos de um objeto de GUI são modificados, uma modificação de tipo C é realizada (4418). FRSs e SRSs são identificadas nas declarações de script de teste transformado 3926 para o objeto de GUI com os atributos modificados e mensagens de guia de mudança 3938 são extraídas. Quando os valores de objetos de GUI são adicionados ou removidos, modificações do tipo D ocorrem (4418). Após a localização de FRSs que referenciam objetos de GUI cujos valores foram mudados, as SRSs são encontradas e o analisador de script determina o impacto devido as SRSs. Quando o tipo de um objeto de GUI é modificado, então, uma modificação do tipo E ocorre, que envolve a localização de FRSs, checar os novos tipos do objeto de GUI, invocar as regras de subsunção de tipo correspondente (por exemplo, regras que a lógica de regras de classe de GUI 3928 pode aplicar) (4418). A lógica de regras de classe de GUI 3928 e/ou o motor de satisfação de restrição 188 pode analisar os objetos de GUI modificados para determinar se erros de CE-1 e CE-1 ocorreram, como resultado da declaração de script de teste transformado 3926 tentar acessar objetos de GUI cujos tipos, propriedades ou valores-padrão são mudados em uma versão

de GAP subsequente 152, e/ou tentar uma operação em um objeto de GUI que não leva em consideração novas restrições impostas nos elementos do objeto de GUI (4420). Em uma implementação, o analisador de script 170 extrai as SRSs identificadas como declarações de script de teste transformado 3926 e mensagens de guia de mudança 3938 correspondentes, de modo que o pessoal de teste possa rever e decidir como modificar as SRSs (4408).

[00326] A Figura 48 mostra um fluxograma para extração de um especificador de mudança de GAP 184. Um modelo de diferença de GUI 162 é produzido (4802) e um analisador de script 170 recebe o modelo de diferença de GUI 162 e os metadados de elemento de GUI 140 (4804). O analisador de script 170 recebe uma representação de script de teste atual 3914 que inclui um vetor de declaração de script de teste 3916 (4806). A lógica de analisador gramatical de script 3912 analisa gramaticalmente o vetor de declaração de script de teste 3916 em nós de vetor para determinar os objetos de GUI para os quais o vetor de declaração de script de teste 3916 navega (4808). A representação de script de teste atual 3914 invoca a OR Lookup 172 para cada objeto de GUI identificado pelo vetor de declaração de script de teste 3916 para recuperação das propriedades dos objetos de GUI a partir do depósito de objeto 174 (4810). A lógica de travessia de caminho 3924 analisa o caminho de navegação das entradas de diferença de elemento de GUI 3910 que corresponde aos objetos de GUI identificados pelo vetor de declaração de script de teste 3916 (4812). A lógica de regras de classe de GUI 3928 analisa as entradas de diferença de elemento de GUI 3910 com base na lógica de regras de classe de GUI 3928 e nas regras de mudança de script de elemento de GUI 3930 para a identificação de entradas de diferença de elemento de GUI válidas 3910 às quais o caminho de navegação corresponde (4814). O motor de satisfação de restrição 188 analisa as

entradas de diferença de elemento de GUI 3910 com base nas regras de satisfação de restrição 3932 para determinar o tipo de especificador de mudança a extrair, incluindo se é para extrair uma declaração de script de teste transformado 3926, uma mensagem de guia de mudança 3938 ou ambas (4816). A lógica de análise de script 3920 extrai um especificador de mudança correspondente ao tipo de mudança de elemento de GUI identificada pela lógica de classe de GUI 3928 e pelas regras de satisfação de restrição 3932 (4818).

[00327] Em uma implementação, a SAA 108 usa uma programação adaptativa incluindo gráficos de classe e de objeto e uma abstração que trata todos os objetos uniformemente. A lógica de travessia 3924, o motor de satisfação de restrição 188 e a lógica de regras de classe de GUI 3928 podem distinguir tipos complexos e simples dos objetos de GUI. Os tipos complexos contêm campos enquanto os tipos simples não. Seja T conjuntos finitos de nomes de tipo e F de nomes de campo ou rótulos, e dois símbolos distintos isto $\in$ F e $\diamondsuit \in$ F. Os gráficos de tipo são gráficos diretos G= (V, E, L), de modo que:

[00328] V $\subseteq$ T, os nós são nomes de tipo;

[00329] L $\subseteq$ F, as bordas são rotuladas por nomes de campo, ou "$\diamondsuit$" onde campos não têm nomes. As bordas que são rotuladas por "$\diamondsuit$" são denominadas bordas de agregação, e bordas que são rotuladas por bordas de referência de nomes de campo. A diferença entre bordas de agregação e de referência se torna clara com o exemplo a seguir. Os campos de classes em linguagens orientadas para objeto designam instâncias de algumas classes, e estes campos têm nomes que são usados para a referência aos campos. Cada campo de uma classe é definido pelo nome do campo e pelo nome da classe (tipo) de que este campo é uma instância. O nome de um campo é o rótulo da borda de referência correspondente no gráfico de

tipo.

[00330] Quando uma classe designa um objeto de GUI o_k e a outra classe designa um objeto de GUI o_n que está contido no objeto o_k, o gráfico de tipo tem dois nós, um para o objeto o_k e o outro para o objeto o_n que o objeto o_k contém. Os nomes das classes correspondentes servem como seus tipos. A relação entre dois objetos sem nome é representada usando-se a borda rotulada com o "◇" no gráfico de tipo.

[00331] $E \subseteq L \times V \times V$, as bordas são produtos vetoriais de rótulos e nós;

[00332] para cada $v \subseteq V$, os rótulos de todas as bordas de saída com exceção de "◇" são distintos;

[00333] para cada $v \subseteq V$, onde v representa um tipo concreto, $v \xrightarrow{\text{este}} v \in E$.

[00334] Um gráfico do objeto é um gráfico dirigido rotulado O = (V', E', L') que é uma instância de um gráfico de tipo G = (V, E, L) sob uma dada função Class que mapeia objetos para suas classes, se as condições a seguir forem satisfeitas:

[00335] para todos os objetos $o \in V'$, o é uma instância do tipo concreto dada pela função Class(o);

[00336] para cada objeto $o \in V'$, os rótulos de suas bordas de referência de saída são exatamente aqueles do conjunto de rótulos de referências de Class(o) incluindo bordas e seus rótulos herdados de classes pais;

[00337] para cada borda $o \xrightarrow{e} o' \in E'$, Class(o) tem uma borda de referência $v \xrightarrow{e} u$ de modo que v seja um tipo pai de Class(o) e u seja um tipo pai de Class(o').

[00338] Um gráfico de objeto é um modelo dos objetos, representados em GAPs, e suas referências a cada outro. Uma

coleção de campos em um gráfico de objeto é um conjunto de bordas rotuladas por nomes de campo. Uma coleção de objetos agregados em um gráfico de objeto é um conjunto de bordas rotuladas por "◇".

Um caminho em um gráfico de tipo G = (V, E, L) é uma sequência de nós e rótulos $p_G = (v_0e_1,v_1e_2, ...e_nv_n)$, onde $v_i \in V$ e $vi \xrightarrow{ei+1} vi+1$ para $0 \leq i \leq n$. Um caminho concreto é definido como uma sequência alternada de nomes de tipo e rótulos designando bordas de referência. Em geral, um caminho A concreto é definido como uma sequência alternativa de nomes de tipo e rótulos designando bordas de referência. Em geral, um caminho concreto pc é um subconjunto do caminho de tipo correspondente p_G, isto é, $p_c \subseteq p_G$.

[00339] Um gráfico de objeto tem o objeto especial $o_r \subseteq V'$, o_r é uma coleção de objetos de raiz $o_r \subseteq V'$ no gráfico de objeto O dado pela função raiz: O→. Este objeto tem Class(o_r) tipo = raiz e sua relação com objetos em sua coleção é expressa através de $o_r \rightarrow \diamond \rightarrow o' \in E'$.

[00340] Dado um objeto o de algum tipo, a lógica de travessia 3924, o motor de satisfação de restrição 188 e a lógica de regras de classe de GUI 3928 podem trabalhar em conjunto para a identificação de um ou mais objetos alcançáveis que satisfaçam a certos critérios. A tarefa realizada é equivalente a determinar se os vetores de declaração de script de teste 3916 e/ou as declarações de script de teste transformado 3926 que descrevem os caminhos de navegação são válidos. Os caminhos de navegação especificados nas declarações de script de teste transformado 3926 podem ser pensados como uma especificação de restrições para o problema de capacidade de alcance de objeto. Encontrar objetos alcançáveis é feito através de travessias. A travessia de uma borda rotulada e corresponde à recuperação do valor do campo e. Toda borda no gráfico de objeto é uma imagem de uma borda que tem parte no gráfico de tipo: há uma borda $e(o_1, o_2)$ em

O apenas quando existirem os tipos v_1 e v_2 de modo que o objeto o_1 seja de tipo v_1, v_1 tenha uma e-parte de tipo v_2, e o_2 seja de tipo v_2.

[00341] O primeiro nó de um caminho p é denominado a origem de p e o último nó é denominado o destino de p. Uma travessia de um gráfico de objeto O começada com um objeto v_i e guiada por caminhos a partir de um conjunto de caminhos p é feita pela realização de uma busca primeiro na profundidade em O com p usado para podar a busca. O histórico de travessia resultante é uma travessia primeiro na profundidade do gráfico de objeto ao longo de caminhos de objeto de acordo com o dado conjunto de caminho concreto.

[00342] O problema de identificação de todos os objetos alcançáveis a partir de um dado objeto o que satisfaz a certos critérios é formalizado conforme se segue. Para cada par de classes c e c', um conjunto de bordas e pode ser identificado pela computação de FIRST(c, c') caso seja possível que um objeto de tipo c alcance um objeto de tipo c' por um caminho começando com uma borda e. Mais precisamente, FIRST(c, c') = e $\in$ E, de modo que exista um gráfico de objeto O de C e objetos o e o' de modo que: 1) Class(o) = c; 2) Class(o') = c'; e 3) o e* o'.

[00343] A última condição, o e* o' indica que existe ($\exists$) um caminho apartir de o para o' no gráfico de objeto, consistindo em uma borda rotulada e, seguido por qualquer sequência de bordas no gráfico. A falta de informação sobre o gráfico real é representada pelo operador de existência $\exists$.

[00344] A tarefa de checagem estática de scripts de teste (por exemplo, scripts de teste transformados 178) é grandemente simplificada quando os nomes de nomes de componentes estranhos são definidos como constantes de string. Quando os nomes de objetos de GUI são especificados usando-se expressões, os valores destas expressões não podem ser determinados até o tempo de execução.

Os gráficos de tipo facilitam o sistema analisador de script 3900 para inferir tipos de expressões e variáveis que mantêm os nomes de objetos de GUI. O sistema analisador de script 3900 aplica conceitos com base na Análise de Gráfico de Travessia (TGA) definida em uma programação adaptativa para inferência de tipos de expressões e variáveis.

[00345] Uma estratégia adaptativa S=(R, π, δ) representa caminhos em um gráfico de objeto, onde R={s, d}, onde s e d são os objetos de GUI de origem e de destino de um caminho em um gravador, e $R \subseteq O$, onde O é o conjunto de objetos em um gráfico de tipo, $\pi = \{e, \alpha\}$, onde e é um conjunto de campos e α é um conjunto de variáveis que designam um conjunto de algumas bordas $\alpha \subseteq e$, e $\delta = \{\rightharpoonup, \rightharpoondown\}$ é um conjunto de variáveis de transição representando objetos e atributos, respectivamente. Cada elemento em uma estratégia S é o nome de algum objeto ou uma variável designando um objeto e/ou atributos.

[00346] A expressão π(o, o') designa um conjunto de objetos {o'}, de modo que cada objeto o' do conjunto seja uma parte do objeto o expresso por alguma borda $e \in \pi$ de modo que e(o, o'). Por exemplo, declarações de script de teste podem ser consideradas estratégias que definem bordas de gráfico de estratégia $a \in b$ e $a \in b$ para declarações de script de teste

[00347] Window("a").VBWindow("b") e

[00348] Window("a").VBWindow("b").property("ReadOnly"),

respectivamente. Assim, uma estratégia é uma abstração de declarações de script de teste (por exemplo, as declarações de script de teste transformado 3926), bem como uma abstração de um conjunto de caminhos em um gráfico de tipo.

[00349] Por exemplo, um gráfico de tipo de uma estrutura organizacional de uma companhia pode incluir: um CEO como um tipo de raiz de algum objeto de GUI que contém o objeto de GUI estoque

de tipo inteiro e agrega o tipo CTO. CTO é um tipo que tem objetos de GUI salário de tipo Cheque (Check) e chefe de tipo CEO. O tipo Cheque por sua vez tem os campos quantidade de tipo flutuante e emissor do tipo CEO. Uma estratégia de quantidade de CEO→ α1 → α2 → para a declaração de script de teste:

[00350] Window("CEO").Window(strexp1).Window(strexp2).property (quantidade) para o gráfico de tipo descrito acima designa a estratégia S, onde s = CEO, d = quantidade, α1 é uma variável que designa objetos computados usando-se a expressão de string strexp1, e α2 é uma variável que designa um objeto de atributo computado através da expressão de string strexp2. Computando-se π(CEO, o') o tipo {CTO} é obtido, e computando-se π(CTO, o') os tipos {CEO,check} são obtidos.

[00351] A cada nó em uma estratégia é atribuído um número de sequência distinto, e os nós são expressos como pares (i, π). Dadas as funções Δi: N× N→ δ e Δπ: π × π → δ e dois números naturais sequenciais k e k+1, a função Δi computa a borda de transição entre nós a que são atribuídos estes números em S, e ∅ se não houver uma borda de transição. Correspondentemente, dados dois nós π_q e π_r em algum gráfico de tipo, a função Δπ computa a borda de transição entre os nós, e ∅ se não houver uma borda de transição.

[00352] Quando os valores de expressões de string em declarações de scripts de teste não podem ser computados até o tempo de execução, as expressões de string podem ser inferidas. A lógica de travessia de caminho 3924, o motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 trabalham em conjunto para a análise das declarações de script de teste transformado, usando gráficos de tipo pela transformação das declarações de script de teste transformado 3926 em uma estratégia adaptativa com variáveis substituindo expressões de string. O motor de satisfação de restrição

188 e/ou a lógica de regra de classe de GUI 3928 computam possíveis valores para cada variável e geram caminhos de travessia para cada estratégia. Quando nenhum caminho é identificado entre os objetos de origem e de destino, então, uma entrada de regra de mudança de script de elemento de GUI 3930, uma mensagem de guia de mudança 3938 e um especificador de mudança 184 podem ser reportados. Quando pelo menos um caminho é identificado, então, uma mensagem de guia de mudança 3938 correspondente e um especificador de mudança 184 são gerados, uma vez que os valores de expressões que computam nomes de objetos podem não estar nos caminhos computados. Em uma implementação, a lógica de travessia de caminho 3924, o motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 podem ser aplicados de forma similar para validação das declarações de script de teste no script de teste atual 164.

[00353] A lógica de travessia de caminho 3924 identifica um ou mais caminhos possíveis, enquanto o motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 validam caminhos para as expressões e declarações. O motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 computam o conjunto de bordas e para cada par de classes c e c', pela computação de FIRST(c, c') onde um objeto de tipo c existe, que pode alcançar um objeto de tipo c' por um caminho começando com uma borda e. Lembre que FIRST(c, c') = e $\in$ E, de modo que exista um gráfico de objeto O de C e objetos o e o' de modo que: 1) Class(o) = c; 2) Class(o') = c'; e 3) o e* o'.

[00354] A última condição, o e* o' indica que existe ($\exists$) um caminho a partir de o para o' no gráfico de objeto, consistindo em uma borda rotulada e, seguido por qualquer sequência de bordas no gráfico. Em uma implementação, o método FIRST é implementado usando-se dois

conjuntos de lógica: a lógica de travessia de caminho 3924 e a lógica de regra de classe de GUI 3928. A Tabela 14 ilustra a lógica de travessia de caminho 3924 e a lógica de regra de classe de GUI 3928.

[00355] A lógica de travessia de caminho 3924 assume o conjunto R de componentes de origem e de destino em S e o conjunto π como parâmetros de entrada. A lógica de travessia de caminho 3924 extrai uma árvore de caminhos válidos em um gráfico de tipo que satisfaz a uma dada estratégia. Alguns dos componentes de entrada podem não fazê-lo na árvore de caminho porque eles não começam quaisquer caminhos válidos.

[00356] Em uma implementação, a lógica de travessia de caminho 3924 invoca a lógica de regra de classe de GUI 3928, a qual por sua vez chama a si mesma de forma recursiva. A lógica de regra de classe de GUI 3928 usa três parâmetros: um componente o que é um nó atual potencial no caminho, o número de sequência i do nó na estratégia S, e a borda de transição δ entre os nós em S a que são atribuídos dois números naturais sequenciais i e i+1. A meta da lógica de regra de classe de GUI 3928 é colorir o nó atual potencial o no caminho como vermelho ou azul, onde um objeto colorido de vermelho o é considerado um nó morto no caminho no gráfico de tipo que não leva aos nós de destino designados. Caso contrário, o nó é colorido de azul e esta cor é propagada até os nós de origem, os quais são subsequentemente incluídos na árvore de caminho.

[00357] A lógica de regra de classe de GUI 3928 se completa quando o número de sequência i é igual a ou maior do que o número de nós na estratégia, |π|, e/ou quando não houver uma borda de transição a partir do nó atual. Quando a lógica de regra de classe de GUI 3928 se completa, a lógica de regra de classe de GUI 3928 colore o nó atual de azul. No procedimento de chamada, a cor do nó é checada, e quando o nó é azul, então, o nó é anexado a seu nó pai na

árvore de caminho.

[00358] Em uma implementação, o motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 trabalham em conjunto para a computação do conjunto de bordas e para cada par de classes c e c', onde um objeto de tipo c é identificado, que pode alcançar um objeto de tipo c' por um caminho começando em uma borda e. A lógica é aplicada individualmente a cada declaração de script de teste transformado 3926 na qual objetos de GAP estranhos sejam especificados usando-se expressões de string cujos valores não sejam conhecidos antes de o script de teste transformado 178 ser executado. O motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 trabalham em conjunto para inferirem possíveis nomes de objetos estranhos que expressões de string possam avaliar no tempo de execução.

Tabela 14 - Travessia de caminho e lógica de regras de classe de GUI

```
      Path Traversal Logic (R ∈ S, π ∈ S)
for all s ∈ R do
   GUI class rules logic (s, 0, Δi(0,1))
   if color(s) = red then
      remove s from R
    end if
end for
GUI class rules logic (o ∈ O, i ∈ N, ∂ ∈ δ)
if i ≥|π| or ∂ = ∈ then
    color(o) ↦blue
else
    for all o' ∈ πi(o, o') do
     if Δπ(o,o') = ∂ then
       GUI class rules logic (o', i + 1, Δi(i, i + 1))
       if color(o') = blue then
       AddChildToTree(o, o')
    end if
  end if
```

```
end for
if children(o) = ∅ then
  color(o) ↦ red
else
    color(o) ↦ blue
    end if
end if
```

[00359] Frequentemente, as mesmas expressões de string são usadas em declarações diferentes no mesmo escopo de script. As mesmas expressões computam os mesmos valores, onde as expressões estão localizadas

[00360] no mesmo escopo, desde que os valores das variáveis usadas nestas expressões não sejam mudados. Usando técnicas de análise de programa, a lógica de travessia de caminho 3924, o motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 trabalham em conjunto para detectarem expressões no tempo de compilação, cujas variáveis não são mudadas no tempo de execução. A lógica de travessia de caminho 3924 identifica um ou mais nomes possíveis de objetos de GUI estranhos que podem ser substituídos por expressões de string em declarações de script de teste. Embora o motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 identifiquem dentre os nomes possíveis de objetos de GUI estranhos, aqueles objetos de GUI que não violam as regras de satisfação de restrição 3932 e/ou as regras de mudança de script de elemento de GUI 194. Dada a mesma expressão usada em declarações de script de teste diferentes no mesmo escopo de script, e desde que os valores das variáveis usadas nestas expressões não sejam mudados por outras expressões executadas entre estas declarações, o motor de satisfação de restrição 188 e/ou a lógica de regra de classe de GUI 3928 identificam um conjunto de nomes de

objetos de GUI estranhos computados por estas expressões de string. Este conjunto de objetos de GUI é obtido tomando-se a interseção dos conjuntos de nomes computados pela lógica de travessia de caminho 3924.

[00361] Por exemplo, considere a quantidade de CEO→ α1 → α2 → do gráfico de estratégia S1 para o gráfico de tipo discutido acima para a expressão de declaração de script de teste transformado 3926: Window("CEO").Window(strexp1).Window(strexp2).property("amount"). O motor de satisfação de restrição 188 e/ou a lógica de regra de classe de GUI 3928 computam valores para as variáveis de esquema de tipo α1= {CTO} e α2= {chefe, salário}.

[00362] Suponha que exista um gráfico de estratégia diferente S2, onde Programmer→ α2 →bonus para y["Programmer"][strexp2].attribute("bonus") para algum outro gráfico de tipo. Note que a variável de expressão de string strexp2 é a mesma em ambas as declarações, e por causa disso a variável de expressão de string strexp2 é designada pelas mesmas variáveis de esquema de tipo em ambos os gráficos de estratégia. Suponha que pela aplicação da lógica de travessia de caminho 3924 valores para a variável de esquema de tipo α2= {salário} sejam computados. Em uma implementação, de modo a se determinar o valor da variável α2 que satisfaz a S1 e a S2, a lógica de regra de classe de GUI 3928 identifica a interseção dos conjuntos de valores de α2 computados para estas duas estratégias. O conjunto resultante α2= {salário} é o resultado de corte dos caminhos de navegação.

[00363] Este exemplo ilustra a idéia de cortar caminhos de navegação usando-se uma análise de fluxo de dados sensível a contexto, que pode ser usada pelo motor de satisfação de restrição 188 e/ou pela lógica de regra de classe de GUI 3928. Pela determinação de definições e usos de uma variável que designa

nomes de objetos de GUI em um dado escopo, os conjuntos de valores são computados para cada declaração de script de teste transformado na qual uma variável é usada. Então, a interseção destes conjuntos é tomada para a determinação dos valores comuns que esta variável pode assumir no escopo considerado.

[00364] O sistema analisador de script 3900 provê uma integridade de modularização como um mecanismo para garantia da capacidade de manutenção de scripts de teste transformados 178. A integridade de modularização especifica que cada declaração em um script de teste transformado 178 possa apenas se comunicar diretamente com objetos que pertençam a GUIs para as quais o script de teste transformado 178 é criado. Composições de scripts de teste transformados 178 nas quais objetos de GUI são acessados pela chamada de funções exportadas por scripts de teste transformados 178 não devem violar a integridade de modularização. O sistema analisador de script 3900 assegura a integridade de modularização de scripts de teste transformados 178 pela análise de composições de declarações de script de teste transformado 3924 para a construção das relações transitivas entre o script de teste atual 164 e o script de teste transformado 178.

[00365] Por exemplo, uma declaração Func("y", "z"), encontrada em uma suíte de scripts de teste relacionados, navega para o campo z de um objeto de GUI estranho y em alguns scripts de teste que exportam a função Func. Assim, alguns scripts de teste na suíte de scripts de teste relacionados podem violar a integridade de modularização ao interoperarem de forma implícita os scripts de teste através da função Func, embora esta comunicação possa ser proibida pelas restrições de uma dada suíte de teste. Em uma implementação, o sistema analisador de script 3900 codifica restrições de modularização quando da definição de scripts de teste usando as restrições de palavra-chave

como parte de um comentário global em cada script de teste. Estas restrições definem GAPs e suas telas de GUI, bem como outros scripts de teste com os quais um dado script de teste pode se comunicar. Um exemplo é uma declaração que especifica uma restrição, que é 'constraints screen("Q") test_scripts("P, S"). Esta restrição efetivamente proíbe um dado script de teste de se comunicar com outros GAPs, telas de GUI e scripts de teste, exceto a tela Q e os scripts de teste P e S, de forma explícita ou implícita. Em uma implementação, o motor de satisfação de restrição 188 assegura que essas restrições não sejam violadas pela manutenção de regras de satisfação de restrição 3932 impostas em scripts de teste e GAPs, e o motor de satisfação de restrição 188 emite mensagens de guia de mudança 3938 quando estas restrições forem violadas.

[00366] A complexidade de tempo da lógica de travessia de caminho 3924, do motor de satisfação de restrição 188 e da lógica de regra de classe de GUI 3928 é exponencial para o tamanho do gráfico de tipo para cada script de teste transformado 178. Devido ao fato de a lógica de travessia de caminho 3924, o motor de satisfação de restrição 188 e a lógica de regra de classe de GUI 3928 envolverem a busca de um ou mais nós e bordas no gráfico de tipo que contenham ciclos para cada nó na estratégia, a complexidade de tempo é $O((V+E)^{\max(|\pi|)})$ onde V é o número de nós, E é o número de bordas no gráfico de tipo, e $\max(|\pi|)$ é o número máximo de nós em estratégias. As operações de armazenamento de sucessores na tabela de variáveis leva $O(1)$. Em geral, o número de nós $\max(|\pi|)$ em estratégias é muito menor do que o número de nós em gráficos de tipo. Nem todos os nós de gráfico podem precisar ser explorados para cada nó em uma estratégia. O limite teórico na complexidade computacional da lógica de travessia de caminho 3924, do motor de satisfação de restrição 188 e da lógica de regra de classe de GUI 3928

é exponencial. Contudo, uma avaliação experimental mostrou que, na prática, o tempo de execução é pequeno para esquemas grandes, porque tipicamente as expressões de caminho são curtas.

[00367] A Figura 49 mostra um analisador de transformação de script de teste com um motor de custo econômico ("arquitetura de motor de custo econômico") 110. Embora descrições detalhadas dos recursos da arquitetura de motor de custo econômico 110 sejam providas adicionalmente abaixo, uma breve introdução da arquitetura de motor de custo econômico 110 será apresentada, primeiramente. Em uma implementação, a arquitetura de motor de custo econômico 110 recebe um modelo de diferença de GUI 162 que especifica diferenças de elemento de GUI entre uma versão de GAP atual 150 e uma versão de GAP subsequente 152. O modelo de diferença de GUI 162 pode ser representado como um esquema em XML. O modelo de diferença de elemento de GUI 162 pode incluir modelos de árvore de GAP subsequentes correspondentes à versão de GAP atual 150 e à versão de GAP subsequente 152. Em uma implementação, o motor de custo econômico 182 recebe o modelo de árvore de GAP atual a partir do modelo de diferença de elemento de GUI 162. Em uma outra implementação, os modelos de árvore de GAP atual e subsequente, bem como o modelo de diferença de GUI 162 são implementados como modelos relacionais armazenados em um banco de dados. A arquitetura de motor de custo econômico 110 emprega uma interface 190 para o recebimento de entradas e comunicação com vários componentes, incluindo o depósito de metadados de elemento de GUI 138. O depósito de metadados de elemento de GUI 138 pode prover uma informação detalhada referente aos elementos de GUI representados no modelo de diferença de GUI 162, o GAP atual 150 e o GAP subsequente 152.

[00368] A arquitetura de motor de custo econômico 110 inclui um

analisador gramatical de script 166 que analisa gramaticalmente um script de teste atual 164 para a obtenção de uma representação intermediária do script de teste atual 164. A representação intermediária pode ser uma árvore de sintaxe abstrata (AST) 168 ou outra representação do script de teste atual 164. Em uma implementação, a arquitetura de motor de custo econômico 110 emprega um motor de custo econômico 182 que analisa a AST 168 e o modelo de diferença de GUI 162. O motor de custo econômico 182 pode invocar a lógica de consulta de depósito de objeto (OR) 172 para a busca em um depósito de objeto 174 e no modelo de diferença de GUI 162 para a localização, no modelo de diferença de GUI 162, dos elementos de GUI identificados pela AST 168. O motor de custo econômico 182 inclui a lógica de modelo de custo econômico 4996 que usa os elementos de GUI identificados pela AST 168 e o modelo de diferença de GUI 162 para a geração de especificadores de mudança de GAP sintéticos 4984 que são usados para se buscar em um depósito de modelos econômicos 176 quanto a uma regra de custo de mudança de elemento de GUI. Em uma implementação, a lógica de modelo de custo econômico 4996 usa os especificadores de mudança de GAP 184 e os especificadores de mudança de GAP sintéticos 4984 para a busca em um depósito de modelos econômicos 176 por uma regra de custo de mudança de elemento de GUI correspondente. O motor de custo econômico 182 aplica cada regra de custo de mudança de elemento de GUI para a obtenção dos custos de transformação de GUI correspondentes a partir do que um relatório de custo de transformação de script de teste 186 é gerado, discutido adicionalmente abaixo. O motor de custo econômico 182 também pode usar medidas de teste históricas a partir de um depósito de medidas de performance 4998 e um depósito de relatórios de custo 5288 para facilitação da obtenção dos custos de transformação de

GUI, discutidos abaixo.

[00369] O motor de custo econômico 182 pode gerar os relatórios de custos de transformação de script de teste 186 com base em combinações diferentes de informação disponível, incluindo: 1) especificadores de mudança de GAP 184 e/ou especificadores de mudança de GAP sintéticos 4984; 2) especificadores de mudança de GAP e um script de teste atual 164; 3) especificadores de mudança de GAP 184, um script de teste atual 164 e uma versão de GAP atual 150 (por exemplo, um modelo de árvore de GAP atual); 4) um script de teste atual 164 e um modelo de diferença de GUI 162 com entradas de diferença de elemento de GUI; e 5) especificadores de mudança de GAP 184, um script de teste atual 164 e um modelo de diferença de GUI 162. Outras combinações da mesma informação ou de uma diferente também podem ser empregadas. As várias combinações de informação disponível são usadas pelo motor de custo econômico 182 para análise dos especificadores de mudança de GAP sintéticos 4984 recebidos e/ou gerados que são usados pela lógica de modelo de custo econômico 4996 para localização e recuperação de custos de transformação de GUI e geração de relatórios de custo de transformação de script de teste.

[00370] A acurácia dos custos de transformação de GUI pode depender, em parte, de quão estreita é a variância entre os custos reais e os custos indicados pelos custos de transformação de GUI. Uma grande variância corresponde a uma acurácia mais baixa, enquanto uma variância estreita corresponde a uma acurácia mais alta. Em outras palavras, a acurácia dos custos de transformação de GUI pode corresponder à previsibilidade e/ou à confiança com que os custos reais refletirão os custos de transformação de GUI. Em uma implementação, a acurácia de custos de transformação de GUI varia devido à granularidade da informação recebida pelo motor de custo

econômico 182. Por exemplo, os custos de transformação de GUI gerados como resultado do motor de custo econômico 182 recebem especificadores de mudança de GAP 184, um script de teste atual 164 e um modelo de diferença de GUI 162 pode ter um nível mais alto de acurácia do que os custos de transformação de GUI gerados unicamente com base em especificadores de mudança de GAP 184. O motor de custo econômico 182 pode empregar vários modelos econômicos que compensem a falta de granularidade de informação provida para o motor de custo econômico 182, discutido em mais detalhes abaixo.

[00371] Em uma implementação, o motor de custo econômico 182 recebe os especificadores de mudança de GAP 184, um script de teste atual 164 e uma versão de GAP atual 150, e gera um modelo de diferença de GUI 162. Em outras palavras, o motor de custo econômico 182 pode gerar o relatório de custo de transformação de script de teste 186 com base na AST 168, na versão de GAP atual 150 e em especificadores de mudança de GAP 184, discutidos adicionalmente abaixo, sem se basear em uma versão de GAP subsequente real 152. Por exemplo, o motor de custo econômico 182 analisa os especificadores de mudança de GAP 184 e a versão de GAP atual 150 (por exemplo, um modelo de árvore de GAP atual recebido a partir do modelo de diferença de GUI 162) e gera os especificadores de mudança de GAP sintéticos 4984. Em uma outra implementação, o motor de custo econômico 182 gera o relatório de custo de transformação de script de teste 186 com base nos especificadores de mudança de GAP 184, sem analisar o modelo de diferença de GUI 162 e/ou a AST 168. O motor de custo econômico 182 pode gerar os relatórios de custo de transformação de script de teste 186 com mensagens de guia de mudança recuperadas a partir de um depósito de mensagem de guia de mudança 192. As

mensagens de guia de mudança podem prover uma informação referente a várias mudanças correspondentes aos especificadores de mudança de GAP 184, a regras de custo de mudança de elemento de GUI e/ou entradas de diferença de GUI.

[00372] A Figura 34 mostra uma GUI de uma versão de GAP subsequente 152. Em uma implementação, o modelo de diferença de GUI 162 resulta de uma comparação entre o modelo de árvore de GAP atual, conforme mostrado na Tabela 6, e um modelo de árvore de GAP subsequente, conforme mostrado na Tabela 8. Em uma outra implementação, o motor de custo econômico 182 analisa o modelo de árvore de GAP atual usando especificadores de mudança de GAP 184 para a determinação de mudança de GAP que podem produzir o modelo de árvore de GAP subsequente, a partir do que o motor de custo econômico 182 gera os especificadores de mudança de GAP sintéticos 4984. Por exemplo, antes de realmente construir a versão de GAP subsequente 152, um programador pode identificar várias mudanças propostas para uma versão de GAP atual 150 usando os especificadores de mudança de GAP 184. O motor de custo econômico 182 analisa os especificadores de mudança de GAP 184 e o modelo de árvore de GAP atual para a geração de especificadores de mudança de GAP sintéticos 4984 que correspondem a um modelo de árvore de GAP subsequente proposto. Em uma implementação, o motor de custo econômico 182 inclui uma lógica de especificador de mudança de GAP que gera os especificadores de mudança de GAP sintéticos 4984 usando o modelo de árvore de GAP atual e os especificadores de mudança de GAP recebidos 184. Em uma implementação, a arquitetura de motor de custo econômico 110 gera os especificadores de mudança de GAP sintéticos 4984 como resultado do registro de várias mudanças propostas identificadas pelo programador durante uma sessão interativa com a versão de GAP

atual 150.

[00373] A Figura 50 mostra um script de teste atual 5000 para a GUI de GAP atual. O script de teste atual 5000 inclui declarações de navegação (por exemplo, L1 e L6) que navegam para objetos de GUI, realizam ações (funções) de leitura, escrita ou outras nos objetos de GUI, e os argumentos destas funções. Por exemplo, a linha 1 do script de teste atual 5000 navega para uma janela StateList, localiza 'Open File' identificado como um objeto de GUI filho da janela StateList, e realiza uma ação de 'Click' no objeto de GUI 'Open File' nas coordenadas 86, 12. Através da série de declarações de navegação e de ação, o script de teste atual 5000 abre um arquivo 'university.data', conforme indicado pelas linhas 2-3. Um 'State' é selecionado a partir de StateListbox e uma SchoolListbox é ativada, com base nas linhas 4-6. O script de teste atual 5000 sucessivamente seleciona a partir da SchoolListbox as escolas 'Acme University' e 'State University' localizadas no 'State' pelo uso de um 'For Next Loop', como resultado das declarações de navegação e de ação nas linhas 7-17, com base nas coordenadas 67, School_Constant em WinObject "Select School" na linha 8. O script de teste atual 5000 usa uma ramificação condicional nas linhas 11-15 para mudança da escala acadêmica para '3' para 'Acme University' e para '2' para 'State University', e salva as mudanças individuais como um resultado da linha 16. O script de teste atual 5000 salva todas as mudanças em um novo arquivo 'university_revise.data' como resultado das declarações nas linhas 18-20.

[00374] A Figura 51 ilustra uma representação de script de teste atual 5100 que o analisador gramatical de script 166 produz como uma representação intermediária do script de teste atual 5000. Em uma implementação, a arquitetura de motor de custo econômico 110 implementa a representação de script de teste atual como uma árvore

de sintaxe abstrata (AST) 168. A representação de script de teste atual 5100 representa um vetor (por exemplo, o script de teste atual 5000) cujos elementos são vetores que representam declarações de navegação do script de teste atual 5000. Em outras palavras, a representação de script de teste atual 5100 representa as declarações de navegação como vetores de declaração de script de teste que navegam para objetos de GUI e realizam ações naqueles objetos de GUI.

[00375] O analisador gramatical de script 166 representa os vetores de declaração de script de teste como uma sequência ordenada de nós que contêm nomes de função e os argumentos daquelas funções que navegam para objetos de GUI. Os nós de um vetor de declaração de script de teste incluem um nó de origem e um destino. Por exemplo, o analisador gramatical de script 166 pode representar o vetor de declaração de script de teste correspondente à linha 1 do script de teste atual 5000 como um nó de origem StateList 5102 e um nó de destino 'Open File' 5104. Os nós de um vetor de declaração de script de teste também podem incluir nós intermediários posicionados entre um nó de origem e um nó de destino. Por exemplo, o analisador gramatical de script 166 pode representar o vetor de declaração de script de teste correspondente à linha 19 do script de teste atual 5000 como um nó de origem StateList 5102, um nó intermediário de 'Save a Data Record' 5106 e um nó de destino 'File name' 5108. O vetor de declaração de script de teste correspondente à linha 19 do script de teste atual 5000 pode ser adicionalmente expresso como (5102-5106-5108-5120-5122), incluindo o método 'Set' 5120 e o valor 'university_revise.data' 5122.

[00376] Os vetores de declaração de script de teste podem incluir nós de entrada em laço e ramificação condicional (por exemplo, o laço 5124 e a ramificação 5126, respectivamente). Em uma

implementação, o nó de laço 5124 é seguido por uma variável de laço (por exemplo, school_constant 5130) para a qual uma faixa de valores, a partir de um limite inferior para um superior (por exemplo, J 5132 e K 5134), são avaliados e usados em expressões dentro do escopo do laço 5124 (por exemplo, coordenadas 67, school_constant 5136). A ramificação 5126 pode ser seguida por uma ou mais condições (por exemplo, condição-1 5138 e condição-2 5140). O motor de custo econômico 182 pode utilizar os nós de laço 5124 e ramificação 5126, e variar os valores e condições (por exemplo, J 5132, K 5134, condição-1 5138 e condição-2 5140) para se avaliar de forma recursiva a representação de script de teste atual 5100.

[00377] Em uma implementação, os especificadores de mudança de GAP 184 incluem um especificador de modelo (discutido em mais detalhes abaixo) que identifica a regra de custo de mudança de elemento de GUI para uso para obtenção de um custo de transformação de GUI correspondente a um laço 5124. A regra de custo de mudança de elemento de GUI para um laço 5124 pode resultar na lógica de modelo de custo econômico 4996 obtendo um custo de transformação de GUI que representa um multiplicador igual ao número de valores na faixa a partir de um limite inferior para um superior (por exemplo, J 5132 e K 5134) que é aplicado aos custos de transformação de GUI para as declarações de script de teste no escopo do laço 5124 (por exemplo, as linhas 8-16 na Figura 50). Por exemplo, o custo de transformação de GUI correspondente ao laço 5124, com base no limite inferior para superior (por exemplo, J 5132 e K 5134), pode ser igual a 10 e os custos de transformação de GUI correspondentes às declarações de script de teste no escopo do laço 5124 (por exemplo, linhas 8-16 na Figura 50) podem equivaler a 500, resultando em um custo de transformação de GUI total de 5.000 para as declarações de script de teste incluindo o laço 5124 (por exemplo,

as linhas 7-17). Em um outro exemplo, o custo de transformação de GUI obtido pela aplicação da regra de custo de elemento de GUI para um laço 5124 pode representar um valor ponderado único (por exemplo, 50) que a lógica de modelo de custo econômico 4996 adiciona aos custos de transformação de GUI correspondentes às declarações de script de teste no escopo de laço 5124 de modo que o custo de transformação de GUI total de 550 resulte para as declarações de script de teste incluindo o laço 5124 (por exemplo, as linhas 7-17).

[00378] A regra de custo de mudança de elemento de GUI para uma ramificação 5126 pode resultar na obtenção de um custo de transformação de GUI que é baseado no número de condições (por exemplo, a condição-1 5138 e a condição-2 5140) dentro do escopo da ramificação 5126, e os custos de transformação de GUI para a ramificação 5126 e as declarações de script de teste no escopo da ramificação 5126 são adicionadas para a obtenção dos custos de transformação de GUI totais. Em uma outra implementação, o custo de transformação de GUI correspondente à ramificação 5126 é um multiplicador que é aplicado aos custos de transformação de GUI correspondendo às declarações de script de teste no escopo da ramificação 5126. Por exemplo, duas condições (por exemplo, a condição-1 5138 e a condição-2 5140) existem no escopo da ramificação 5126, correspondendo a um custo de transformação de GUI de 2 para a ramificação 5126 e os custos de transformação de GUI das linhas dentro do escopo da ramificação 5126 são 100 resultando em um custo de transformação de GUI total de 200 para as declarações de script de teste incluindo a ramificação 5126 (por exemplo, as linhas 11-15).

[00379] O analisador gramatical de script 166 avalia argumentos de funções de navegação e de ação como expressões, variáveis e

constantes. Os argumentos expressam as propriedades físicas de objetos de GUI para as quais os vetores de declaração de script de teste navegam e valores usados para a realização das ações naqueles objetos de GUI. Por exemplo, as coordenadas '86, 12' 5112 identificam a localização de um dispositivo de apontar para a realização de uma ação 'Click' 5110 no objeto de GUI 'Open File' 5104, o qual é um objeto de GUI filho da janela StateList 5102. O motor de custo econômico 182 usa os nomes dos objetos de GUI (por exemplo, StateList 5102 e 'Open File' 5104) navegados para pelos vetores de declaração de script de teste para localização das propriedades físicas correspondentes dos objetos de GUI armazenados em um depósito de objeto 174, para a identificação das entradas de diferença de GUI correspondentes e para a geração de especificadores de mudança de GAP sintéticos 4984.

[00380] Em uma implementação, o motor de custo econômico 182 usa a lógica de consulta de OR 172 para localização, no depósito de objeto 174, das propriedades físicas dos objetos de GUI navegados por um vetor de declaração de script de teste, e para localização, no modelo de diferença de GUI 162, de entradas de diferença de GUI correspondentes. O motor de custo econômico 182 gera especificadores de mudança de GAP sintéticos 4984 e invoca a lógica de modelo de custo econômico 4996 para a localização das regras de custo de mudança de elemento de GUI correspondentes no depósito de modelos econômicos 176 usando-se especificadores de mudança de GAP 184 e especificadores de mudança de GAP sintéticos 4984. Em uma IP, a lógica de consulta de OR 172 é dividida em duas subfunções: 1) uma lógica de consulta adaptada para a localização e a recuperação das propriedades físicas dos objetos de GUI navegados pelo vetor de declaração de script de teste (por exemplo, 5102-5104, 5102-5106-5108, e 5102-5114); e 2) uma lógica de localizador que

encontra e retorna uma entrada de diferença de elemento de GUI (nó) no modelo de diferença de GUI 162 que corresponde ao objeto de GUI com as dadas propriedades físicas. A lógica de modelo de custo econômico 4996 gera especificadores de mudança de GAP sintéticos 4984 que são usados para a localização das regras de custo de mudança de elemento de GUI aplicáveis, com base nos resultados de lógica de consulta e de lógica de localizador a partir da lógica de consulta de OR 172. A lógica de consulta de OR 172 pode incluir uma lógica de travessia de caminho, discutida em mais detalhes abaixo, para a identificação de caminhos de navegação possíveis de um vetor de declaração de script de teste entre um objeto de GUI de nó de origem e um objeto de GUI de nó de destino para o qual um vetor de declaração de script de teste navega.

[00381] O motor de custo econômico 182 pode consultar o depósito de objeto 174 para identificar as propriedades físicas dos objetos de GUI navegados pelos vetores de declaração de script de teste representados pela representação de script de teste atual 5100. As propriedades físicas de um objeto de GUI podem indicar se o objeto de GUI está oculto, é apenas de leitura, é um número e valores-padrão, conforme mostrado na Tabela 12.

[00382] Por exemplo, o motor de custo econômico 182 analisa os objetos de GUI 5102-5114 no vetor de declaração de script de teste. A coordenada '19,22' 5118 identifica a localização para um dispositivo de apontar para a realização de uma ação de 'Click' 5116 no objeto de GUI SchooListbox 5114, o qual é um objeto de GUI filho da janela StateList 5102. O motor de custo econômico 182 invoca a lógica de consulta de OR 172 para localização das propriedades físicas dos objetos de GUI 5102 e 5114. A lógica de consulta de OR 172 localiza as propriedades físicas da janela StateList 5102 e do WinObject SchoolListbox 5114, conforme mostrado na Tabela 11 nas linhas 3 e

12. O motor de custo econômico 182 usa as propriedades físicas recuperadas a partir do depósito de objeto 174 para a localização das entradas de diferença de GUI correspondentes (por exemplo, 4504 e 4604) no modelo de diferença de GUI 162. As entradas de diferença de GUI 4504 e 4604 indicam que a janela StateList 5102 e o WinObject SchoolListbox 5114 na versão de GAP atual 150 correspondem à janela School 3402 e ao WinObject SchoolCombobox 3406 na versão de GAP subsequente 152, respectivamente. Em uma implementação, o motor de custo econômico 182 emprega a lógica de consulta de OR 172 para atravessar o modelo de diferença de GUI 162 usando as propriedades físicas dos objetos de GUI navegados pelo vetor de declaração de script de teste. A função da lógica de consulta de OR 172 retorna uma entrada de diferença de elemento de GUI (por exemplo, 3604, 4504 e 4604) a partir do modelo de diferença de GUI 162 que representa o objeto de GUI navegado pelo vetor de declaração de script de teste (por exemplo, 5102-5104-5110-5112, 5102-5106-5108-5120-5122, e 5102-5114-5126-5118), e a lógica de modelo de custo econômico 4996 gera especificadores de mudança de GAP sintéticos correspondentes 4984. A Tabela 12 ilustra as propriedades físicas que podem ser localizadas no depósito de objeto para a entrada de objeto de GUI correspondente à SchoolListbox 5114.

[00383] A Figura 52 mostra um sistema de motor de custo econômico de exemplo 5200 que pode implementar o motor de custo econômico 182. O motor de custo econômico 182 inclui uma memória 5202 acoplada a um processador 5204 e uma interface 190. Em uma implementação, a interface 190 se comunica com o depósito de metadados de elemento de GUI 138 e o modelo de diferença de GUI 162 para receber metadados de elemento de GUI 140 e entradas de diferença de elemento de GUI 3910, respectivamente. A interface 190

é conectada a uma rede 3906 (por exemplo, a Internet) em comunicação com vários outros sistemas e recursos. Em uma outra implementação, a memória 5202 inclui os metadados de elemento de GUI 140, e a lógica de especificador de mudança de GAP 5290 que produz o modelo de diferença de GUI 162 e as entradas de diferença de elemento de GUI 3910. A memória 5202 também inclui a lógica de analisador gramatical de script 3912 que recebe o script de teste atual 164 e produz a AST 168, processa a AST 168 como uma representação de script de teste atual 3914 e produz os vetores de declaração de script de teste 3916 (por exemplo, 5102-5104-5110-5112, 5102-5106-5108-5120-5122, e 5102-5114-5126-5118).

[00384] A memória 5202 ainda inclui a lógica de modelo de custo econômico 4996 que, em uma implementação, invoca a lógica de consulta de OR 172 para a localização, no depósito de objeto 174, de uma entrada de objeto de GUI 3922 referida pelo vetor de declaração de script de teste 3916. Em uma outra implementação, o motor de custo econômico 182 invoca a lógica de consulta de OR 172 para localizar, em várias fontes externas, uma entrada de objeto de GUI 3922 combinando com o vetor de declaração de script de teste 3916. Quando o vetor de declaração de script de teste 3916 (por exemplo, 5102-5104-5110-5112, 5102-5106-5108-5120-5122, e 5102-5114-5126-5118) emprega constantes para a identificação de nomes de objeto de GUI, ao invés de expressões cujos valores podem ser determinados apenas no tempo de execução, a função de lógica de consulta de OR 172 pode usar o nome de objeto de GUI e as propriedades do objeto de GUI para a localização eficiente da entrada de objeto de GUI correta 3922 e para localizar, no modelo de diferença de GUI 162, uma entrada de diferença de elemento de GUI 3910 combinando com a entrada de objeto de GUI 3922.

[00385] Por exemplo, o vetor de declaração de script de teste

representado por 5102-5104-5110-5112 identifica o objeto de GUI de janela StateList 3302 e o objeto de GUI de caixa de lista SchoolListbox 3316, mostrados na declaração de navegação de modelo de diferença de GUI 162 mostrada na linha 6 da Figura 50:

[00386] Window("StateList").WinObject("SchoolListbox").Click 19,22.

[00387] A lógica de consulta de OR 172 localiza cada entrada de objeto de GUI 3922 para os objetos de GUI 3302 e 3316, usando os nomes conhecidos dos objetos de GUI, StateList e SchoolListbox, respectivamente. A lógica de consulta de OR 172 localiza as entradas de diferença de elemento de GUI correspondentes 4504 e 4604 no modelo de diferença de GUI 162. Em uma implementação, a lógica de modelo de custo econômico 4996 analisa as entradas de diferença de elemento de GUI correspondentes 4504 e 4604 e gera um ou mais especificadores de mudança de GAP sintéticos correspondentes 4984. Usando os especificadores de mudança de GAP 184, a lógica de modelo de custo econômico 4996 localiza, no depósito de modelos econômicos 176 uma ou mais regras de custo de mudança de elemento de GUI 5246 e aplica as regras de custo para a obtenção dos custos de transformação de GUI 5248. Os custos de transformação de GUI 5248 podem incluir custos correspondentes à mudança do script de teste atual 164 e a testes da versão de GAP subsequente 152. Em outras palavras, a lógica de modelo de custo econômico 4996 determina os custos para a geração de uma declaração de script de teste transformado que corresponde aos objetos de GUI School 3402 e SchoolCombobox 3406 e os custos para se testar a versão de GAP subsequente 152 usando a declaração de script de teste transformado:

[00388] Window("School").WinObject("SchoolCombobox").Click 294,14.

[00389] Cada regra de custo de mudança de elemento de GUI 5246 pode incluir vários atributos, incluindo: um identificador de especificador de mudança 5250, identificadores de utilização de recurso de sistema 5252, uma estimativa de custo de mudança de GUI 5254 que indica o tempo estimado e/ou recursos necessários para se testar uma mudança de elemento de GUI correspondente, identificadores dependentes de especificador de mudança 5256, uma classificação de dependência 5258, uma classificação de qualidade 5260, uma classificação de complexidade 5262 e custos dependentes de mudança de elemento de GUI 5264. Cada modelo econômico residente no depósito de modelos econômicos 176 e/ou externo ao depósito de modelos econômicos 176 que esteja disponível para a lógica de modelo de custo econômico 4996 através da interface 190 pode incluir mais, menos ou diferentes atributos de regra de custo de mudança de elemento de GUI 5246.

[00390] A lógica de modelo de custo econômico 4996 usa os especificadores de mudança de GAP 184 e especificadores de mudança de GAP sintéticos 4984 para a localização de regras de custo de mudança de elemento de GUI 5246 no depósito de modelos econômicos 176 correspondentes aos identificadores de especificador de mudança 5250. Os identificadores de utilização de recurso de sistema 5252 indicam os recursos usados para se testar uma mudança de elemento de GUI em particular. Em uma implementação, os identificadores de utilização de recurso de sistema 5252 têm valores de 1 a 10 que identificam a quantidade de uma capacidade de processamento e de infraestrutura de ambiente de teste necessária para se testar uma mudança de GAP correspondente. Por exemplo, um identificador de utilização de recurso de sistema 5252 com um valor de 3, correspondente à capacidade de processamento de ambiente de teste, pode indicar que um terço da capacidade de

processamento disponível é necessário para se testar uma mudança de GAP correspondente. Um identificador de utilização de recurso de sistema 5252 com um valor de 10 correspondente à capacidade de processamento de ambiente de teste pode indicar que a maioria dos recursos de computação disponíveis (por exemplo, capacidade de processamento) será necessária para se testar uma mudança de GAP correspondente.

[00391] Em uma outra implementação, os identificadores de utilização de recurso de sistema 5252 provêm descrições dos recursos necessários para se testar uma mudança de GAP correspondente. Por exemplo, os identificadores de utilização de recurso de sistema 5252 podem itemizar as habilidades de testadores, componentes de sistema (por exemplo, dispositivos de entrada e de saída, e largura de banda de rede) e regulagens de prioridade a serem atribuídas aos processos de sistema usados para se testar uma mudança de elemento de GUI correspondente. Em uma implementação, os identificadores de utilização de recurso de sistema 5252 provêm uma combinação de valores discretos que indicam a capacidade de processamento de ambiente de teste e as descrições itemizadas dos vários recursos necessários para se testar a mudança de elemento de GUI correspondente.

[00392] A lógica de modelo de custo econômico 4996 pode localizar outras regras de custo de mudança de elemento de GUI 5246 que dependem de uma regra de custo de mudança de elemento de GUI 5246 identificada por um identificador de especificador de mudança 5250, usando os identificadores dependentes de especificador de mudança 5256. Os identificadores dependentes de especificador de mudança 5256 podem identificar uma ou mais regras de custo de mudança de elemento de GUI 5246 que dependem da mudança de elemento de GUI correspondente ao identificador de especificador de

mudança 5250. Por exemplo, uma mudança na classe de um objeto de GUI pai de uma caixa de lista para uma caixa de combinação (por exemplo, SchoolListbox 3316 e SchoolCombobox 3406) pode impor mudanças de elemento de GUI a objetos de GUI filhos do objeto de GUI pai, de modo que um identificador de especificador de mudança 5250 correspondente a um especificador de mudança de GAP 184 em particular e/ou um especificador de mudança de GAP sintético 4984 identifique um ou mais identificadores dependentes de especificador de mudança 5256.

[00393] Em uma implementação, a classificação de dependência 5258 é um valor de 0 a 10 que indica o nível de dependência que um GAP pode ter em um elemento de GUI em particular. A classificação de dependência 5258 pode corresponder à visibilidade e ao escopo de um elemento de GUI. Por exemplo, a mudança da janela Statelist 3302 no GAP atual 150 para School 3402 no GAP subsequente 152, conforme mostrado na Figura 14, pode corresponder a uma classificação de dependência 5258 de 10, enquanto a mudança de um valor na StateListbox 3312 para um valor na StateListbox 3404, conforme mostrado na Figura 5, pode corresponder a uma classificação de dependência 5258 de 4. A lógica de modelo de custo econômico 4996 usa a classificação de dependência 5258 para facilitar a obtenção dos custos de transformação de GUI 5248. Em uma implementação, a lógica de modelo de custo econômico 4996 usa a classificação de dependência 5258 para determinar como e/ou se é para usar o fator de eficiência de mudança de GUI 5286, discutido em mais detalhes abaixo.

[00394] Em uma implementação, a classificação de qualidade 5260 é um valor de 1 a 10 que indica a contribuição para a qualidade da versão de GAP subsequente 152 feita pela mudança de elemento de GUI. Por exemplo, uma mudança de elemento de GUI em particular

que cumpre uma checagem de integridade em um elemento de GUI correspondente a um valor alto de classificação de dependência 5258 pode corresponder a uma classificação de qualidade 5260 de 10. Em um outro exemplo, uma mudança de elemento de GUI que é imperceptível e/ou corresponde a um valor baixo de classificação de dependência 5258 pode corresponder a uma classificação de qualidade 5260 de 0. Em uma implementação, a lógica de modelo de custo econômico 4996 usa um identificador de preferência de qualidade selecionável para a geração do relatório de custo de transformação de script de teste 186 com custos de transformação de GUI 5248 correspondentes a classificações de qualidade 5260 se adequando ou excedendo ao identificador de preferência de qualidade, de modo que planos de teste possam ser avaliados com base em fatores de qualidade.

[00395] Em uma implementação, a classificação de complexidade 5262 é um valor de 1 a 10 que indica o nível de dificuldade de se testar a mudança de elemento de GUI correspondente, onde um nível de complexidade de 10 é um nível alto de complexidade e um nível de 0 é um nível baixo de complexidade. Em uma outra implementação, a classificação de complexidade 5262 indica a contribuição para o nível de complexidade da versão de GAP subsequente 152 feita pela mudança de elemento de GUI. Por exemplo, uma mudança de elemento de GUI em particular que cumpra uma checagem de integridade em um elemento de GUI correspondente a um valor alto de classificação de dependência 5258 pode corresponder a uma classificação de complexidade 5262 de 10. Em um outro exemplo, uma mudança de elemento de GUI que seja imperceptível e/ou corresponda a um valor baixo de classificação de dependência 5258 pode corresponder a uma classificação de complexidade 5262 de 0. Em uma implementação, a lógica de modelo de custo econômico 4996

usa um identificador de preferência de complexidade selecionável por usuário para a geração do relatório de custo de transformação de script de teste 186 com os custos de transformação de GUI 5248 para classificações de complexidade 5262 se adequando ou excedendo ao identificador de preferência de complexidade, de modo que planos de teste possam ser avaliados com base na complexidade.

[00396] Os custos dependentes de mudança de elemento de GUI 5264 podem representar um custo de transformação de GUI agregado 5248 correspondente aos identificadores dependentes de especificador de mudança 5256. Em uma implementação, a lógica de modelo de custo econômico 4996 usa os custos dependentes de mudança de elemento de GUI 5264, ao invés de recuperar cada regra de custo de mudança de elemento de GUI 5246 correspondente a um ou mais identificadores dependentes de especificador de mudança 5256 para a geração do relatório de custo de transformação de script de teste 186.

[00397] Em uma implementação, a lógica de modelo de custo econômico 4996 usa os identificadores de preferência selecionáveis por usuário 5266 para a localização das regras de custo de mudança de elemento de GUI 5246 com base em valores discretos e/ou faixas de valores para um ou mais dos identificadores de utilização de recurso de sistema 5252, a estimativa de custo de mudança de GUI 5254, a classificação de dependência 5258, a classificação de qualidade 5260, a classificação de complexidade 5262 e os custos dependentes de mudança de elemento de GUI 5264. Os identificadores de preferência 5266 identificam os custos de transformação de GUI 5248 usados para a geração do relatório de custo de transformação de script de teste 186, com base em um ou mais dentre o identificador de especificador de mudança 5250, um identificador de utilização de recurso de sistema 5252, uma estimativa

de custo de mudança de GUI 5254 que indica o tempo estimado e/ou recursos (por exemplo, dinheiro e trabalho) para se testar uma mudança de elemento de GUI correspondente, identificadores dependentes de especificador de mudança 5256, classificação de dependência 5258, classificação de qualidade 5260, classificação de complexidade 5262 e custos dependentes de mudança de elemento de GUI 5264.

[00398] O custo de transformação de GUI 5248 pode incluir um componente de tempo e um componente de recurso. O componente de tempo do custo de transformação de GUI 5248 pode indicar o tempo decorrido necessário para a mudança de uma declaração de script de teste e/ou para se testar uma mudança de elemento de GUI correspondente. O componente de recurso do custo de transformação de GUI 5248 pode indicar o dinheiro, áreas de habilidade e/ou infraestrutura de sistema (por exemplo, unidades humanas e tecnológicas) necessárias para a mudança de uma declaração de script de teste e/ou para se testar uma mudança de elemento de GUI correspondente.

[00399] Lembre que o motor de custo econômico 182 pode gerar relatórios de custo de transformação de script de teste 186 com base em múltiplas combinações de informação disponível, incluindo: 1) especificadores de mudança de GAP 184; 2) especificadores de mudança de GAP e um script de teste atual 164; 3) especificadores de mudança de GAP 184, um script de teste atual 164 e uma versão de GAP atual 150 (por exemplo, um modelo de árvore de GAP atual); 4) um script de teste atual 164 e um modelo de diferença de GUI 162 com entradas de diferença de elemento de GUI; e 5) especificadores de mudança de GAP 184, um script de teste atual 164 e um modelo de diferença de GUI 162. As várias combinações de informação disponível são usadas pelo motor de custo econômico 182 para

análise dos especificadores de mudança de GAP recebidos 184 e/ou dos especificadores de mudança de GAP sintéticos 4984 gerados que são usados pela lógica de modelo de custo econômico 4996 para localização e recuperação de custos de transformação de GUI 5248 e geração de relatórios de custo de transformação de script de teste 186.

[00400] Em uma implementação, o motor de custo econômico 182 recebe especificadores de mudança de GAP 184 que o motor de custo econômico 182 usa para localizar e recuperar os custos de transformação de GUI 5248 a partir do depósito de modelos econômicos 176. Os especificadores de mudança de GAP recebidos 184 podem ter resultado de uma análise prévia de uma versão de GAP atual 150, um script de teste atual 164 e/ou um modelo de diferença de GUI 162. Em uma implementação, o motor de custo econômico 182 pode receber especificadores de mudança de GAP 184 e um script de teste atual 164. A lógica de analisador gramatical de script 3912 produz os vetores de declaração de script de teste 3916 com base no script de teste atual 164. A lógica de modelo de custo econômico 4996 analisa os vetores de declaração de script de teste 3916 para a geração de especificadores de mudança de GAP sintéticos 4984. A lógica de modelo de custo econômico 4996 usa os especificadores de mudança de GAP recebidos 184 e os especificadores de mudança de GAP sintéticos gerados 4984 para a localização e a recuperação de custos de transformação de GUI 5248 a partir do depósito de modelos econômicos 176. Tem uma implementação, os especificadores de mudança de GAP 184 recebidos e os especificadores de mudança de GAP sintéticos gerados 4984 são indistinguíveis quanto a sua origem, de modo que a lógica de modelo de custo econômico 4996 processe os especificadores de mudança de GAP 184 recebidos e os especificadores de mudança de GAP sintéticos gerados 4984

uniformemente, independentemente de sua origem.

[00401] Em uma outra implementação, o motor de custo econômico 182 recebe especificadores de mudança de GAP 184, um script de teste atual 164 e uma versão de GAP atual 150. O motor de custo econômico 182 analisa os especificadores de mudança de GAP 184 e a versão de GAP atual 150 (por exemplo, um modelo de árvore de GAP atual) para a geração de especificadores de mudança de GAP sintéticos 4984. O motor de custo econômico 182 analisa os vetores de declaração de script de teste 3916 correspondentes ao script de teste atual 164 e ao modelo de diferença de GUI 162 para a geração dos especificadores de mudança de GAP sintéticos 4984. A lógica de modelo de custo econômico 4996 usa os especificadores de mudança de GAP 184 recebidos e os especificadores de mudança de GAP sintéticos gerados 4984 para a localização e a recuperação de custos de transformação de GUI 5248 a partir do depósito de modelos econômicos 176 e gera o relatório de custo de transformação de script de teste 186.

[00402] Em uma implementação, o motor de custo econômico 182 recebe um script de teste atual 164 e um modelo de diferença de GUI 162, sem especificadores de mudança de GAP 184. O motor de custo econômico 182 analisa os vetores de declaração de script de teste 3916 correspondentes ao script de teste atual 164 e ao modelo de diferença de GUI 162 para a geração dos especificadores de mudança de GAP sintéticos 4984.

[00403] Em uma implementação, o motor de custo econômico 182 recebe os especificadores de mudança de GAP 184, um script de teste atual 164, um modelo de diferença de GUI 162 correspondente à versão de GAP atual 150 e uma versão de GAP subsequente 152. O motor de custo econômico 182 analisa os vetores de declaração de script de teste 3916 correspondentes ao script de teste atual 164 e ao

modelo de diferença de GUI 162 para a geração de especificadores de mudança de GAP sintéticos 4984. O motor de custo econômico 182 usa os especificadores de mudança de GAP 184 recebidos e os especificadores de mudança de GAP sintéticos gerados 4984 para a localização e a recuperação das regras de custos de transformação de GUI 5248 a partir do depósito de modelos econômicos 176 e para a geração do relatório de custo de transformação de script de teste 186.

[00404] Em uma implementação, a acurácia dos custos de transformação de GUI 5248 varia devido à granularidade da informação recebida pelo motor de custo econômico 182. Por exemplo, os custos de transformação de GUI 5248 gerados como resultado do motor de custo econômico 182 recebe os especificadores de mudança de GAP 184, um script de teste atual 164 e um modelo de diferença de GUI 162 podem ter um nível de acurácia mais alto do que os custos de transformação de GUI 5248 gerados com base unicamente nos especificadores de mudança de GAP 184 recebidos. O motor de custo econômico 182 pode empregar vários modelos econômicos para conservação da acurácia dos custos de transformação de GUI 5248 e compensação de granularidades variáveis de informação provida para o motor de custo econômico 182.

[00405] Em uma implementação, os especificadores de mudança de GAP 184 e/ou os especificadores de mudança de GAP sintéticos 4984 incluem um especificador de modelo 5268, uma frequência de mudança de GUI 5270, coeficientes de habilidade 5272, um identificador de complexidade 5274, um identificador de qualidade 5276, uma percentagem de mudança 5278, um tipo de apagamento de percurso errado 5280, um tipo de mesmo percurso errado 5282 e especificadores de tipo de elemento mudado 5284. O especificador de modelo 5268 especifica um ou mais modelos econômicos a serem usados dentre os múltiplos modelos econômicos acessíveis pela lógica

de modelo de custo econômico 4996. Em uma implementação, o especificador de modelo 5268 especifica um ou mais modelos econômicos para a lógica de modelo de custo econômico 4996 usar correspondentes às granularidades variáveis de informação providas para o motor de custo econômico 182, de modo que a acurácia dos custos de transformação de GUI 5248 seja conservada. Por exemplo, o especificador de modelo 5268 pode especificar modelos correspondentes a uma ou mais das múltiplas combinações de informação disponível recebidas pelo motor de custo econômico 182, incluindo: 1) modelo-1 para especificadores de mudança de GAP 184; 2) modelo-2 para especificadores de mudança de GAP 184 e um script de teste atual 164; 3) modelo-3 para especificadores de mudança de GAP 184, um script de teste atual 164 e uma versão de GAP atual 150; 4) modelo-4 para um script de teste atual 164 e um modelo de diferença de GUI 162 com entradas de diferença de elemento de GUI 3910; e 5) modelo-5 para especificadores de mudança de GAP 184, um script de teste atual 164 e um modelo de diferença de GUI 162 com entradas de diferença de elemento de GUI 3910.

[00406] A frequência de mudança de GUI 5270 indica o número de ocorrências de uma mudança de elemento de GUI em particular. Em uma implementação, a lógica de modelo de custo econômico 4996 inclui um fator de eficiência de mudança de GUI ajustável 5286 que indica se uma frequência de mudança de GUI 5270 acima de um limite em particular resulta em um custo de transformação de GUI mais baixo 5248. Por exemplo, um fator de eficiência de mudança de GUI 5286 de 0,50 indica que o custo de transformação de GUI 5248 para cada mudança acima de um limite de 100 ocorrências para uma dada mudança de elemento de GUI é ajustado em 50 por cento. Em outras palavras, quando uma mudança de elemento de GUI em particular é identificada como tendo 120 ocorrências, a lógica de modelo de custo

econômico 186 aplica o fator de eficiência de mudança de GUI 5286 de 0,50 ao custo de transformação de GUI 5248 para as 20 mudanças acima do limite de 100. Em um outro exemplo, um fator de eficiência de mudança de GUI 5286 de 0,00 pode indicar que nenhuma eficiência é realizada, independentemente do valor de frequência de mudança de GUI 5270.

[00407] Em uma implementação, os coeficientes de habilidade 5272 incluem um ou mais coeficientes que são usados para a descrição do nível de experiência dos testadores que se espera que testem a versão de GAP subsequente 152. Os coeficientes de habilidade 5272 podem incluir coeficientes individuais para áreas específicas de experiência de teste. Por exemplo, os coeficientes de habilidade 5272 podem corresponder à habilidade e ao nível de experiência de testadores de acordo com fases em particular de testes, tais como unidade, integração, sistema e fase de teste final, de modo que cada fase seja representada por um ou mais coeficientes. Em um outro exemplo, os coeficientes de habilidade 5272 podem corresponder às habilidades e à experiência correspondente a aspectos em particular de testes da versão de GAP subsequente 152, tais como segurança e autenticação de usuário, computações numéricas específicas para o GAP, e rede e infraestrutura.

[00408] Em uma outra implementação, os coeficientes de habilidade 5272 são calibrados com base em medidas de performance localizadas em um depósito de medidas de performance 4998 e/ou um depósito de relatórios de custo 5288. Os especificadores de mudança de GAP 184 e/ou especificadores de mudança de GAP sintéticos 4984 podem ser construídos e/ou gerados a partir de medidas de performance históricas encontradas em um depósito de medidas de performance 4998 e/ou um depósito de relatórios de custo 5288. Os coeficientes de habilidades 5272 dos especificadores de mudança de

GAP 184 e/ou especificadores de mudança de GAP sintéticos 4984 construídos podem ser ajustados por múltiplas interações para a obtenção de custos de transformação de GUI 5248 e relatórios de custo de transformação de script de teste 186 que estão em margens aceitáveis de variância para os custos reais refletidos no depósito de medidas de performance 4998. A acurácia dos custos de transformação de GUI 5248 obtidos pela lógica de modelo de custo econômico 4996 pode ser com base em quão bem os coeficientes de habilidades 5272 são calibrados para refletirem os recursos de teste disponíveis para se testar a versão de GAP subsequente 152. Em uma implementação, os coeficientes de habilidades 5272 influenciam o identificador de complexidade 5274, discutido em mais detalhes abaixo.

[00409] A lógica de modelo de custo econômico 4996 usa os coeficientes de habilidades 5272 para a obtenção de custos de transformação de GUI 5248. Por exemplo, um valor de coeficiente de habilidades 5272 de 1,0 pode indicar que é esperado que testadores com pouca experiência sejam usados para teste da versão de GAP subsequente 152 e custos de transformação de GUI 5248 mais altos possam resultar para refletirem a experiência baixa. Em um outro exemplo, um valor de coeficiente de habilidades 5272 de 8,0 pode indicar testadores com uma experiência de teste mais alta do que a média e custos de transformação de GUI 5248 mais baixos podem resultar, que refletem a experiência mais alta do que a média. A lógica de modelo de custo econômico 4996 pode analisar se os coeficientes de habilidades 5272 e a classificação de complexidade 5262 se correlacionam, e obter de forma correspondente custos de transformação de GUI 5248 mais altos ou mais baixos. Por exemplo, os coeficientes de habilidades 5272 podem indicar que os testadores são capazes de testarem um GAP com um nível em particular de

complexidade, conforme indicado pela classificação de complexidade 5262, de modo que custos de transformação de GUI 5248 mais baixos sejam obtidos. Em um outro exemplo, os coeficientes de habilidades 5272 podem indicar que os testadores carecem de uma habilidade e um nível de experiência para testar um GAP com um nível de complexidade em particular correspondente à classificação de complexidade 5262, de modo que custos de transformação de GUI 5248 mais altos sejam obtidos para refletirem a falta de habilidades e experiência dos testadores e o tempo esperado e recursos para se testar a versão de GAP subsequente 152.

[00410] Em uma implementação, o identificador de complexidade 5274 numericamente identifica o nível de complexidade de uma mudança de elemento de GUI (por exemplo, valores de 0 a 10) determinado pela lógica de modelo de custo econômico 4996, correspondente a um especificador de mudança de GAP sintético gerado 4984. Em uma outra implementação, o identificador de complexidade 5274 identifica o nível de complexidade determinado por um testador e recebido pela lógica de modelo de custo econômico 4996 com o especificador de mudança de GAP 184. Distinguindo o identificador de complexidade 5274 do especificador de mudança de GAP 184 e/ou especificador de mudança de GAP sintético 4984 a partir da classificação de dependência 5258 da regra de custo de mudança de elemento de GUI 5246, o identificador de complexidade 5274 representa uma análise que é externa ao depósito de modelos econômicos 176. A lógica de modelo de custo econômico 4996 pode analisar a classificação de complexidade 5262 e o identificador de complexidade 5274 para avaliação da acurácia dos custos de transformação de GUI 5248 obtidos pela aplicação da regra de custo de mudança de elemento de GUI 5246.

[00411] Por exemplo, a lógica de modelo de custo econômico 4996

pode determinar que a classificação de complexidade 5262 e o identificador de complexidade 5274 correspondentes a uma mudança de elemento de GUI em particular estão em uma margem aceitável de variância, de modo que o custo de transformação de GUI 5248 não seja ajustado, como resultado. Em um outro exemplo, a lógica de modelo de custo econômico 4996 pode determinar que a classificação de complexidade 5262 e o identificador de complexidade 5274 correspondentes a uma mudança de elemento de GUI em particular estão fora de uma margem aceitável de variância e os custos de transformação de GUI 5248 são ajustados para cima por um multiplicador. A margem de variância e o multiplicador, determinados pela análise da classificação de complexidade 5262 e do identificador de complexidade 5274, podem ser selecionáveis por usuário e/ou ajustáveis. Em uma implementação, o identificador de complexidade 5274 é com base nos coeficientes de habilidades 5272, de modo que a complexidade de uma mudança de elemento de GUI seja avaliada em relação às habilidades e à experiência dos testadores disponíveis. Os coeficientes de habilidades 5272 podem ser calibrados de modo que a classificação de complexidade 5262 e o identificador de complexidade 5274 gerados pela lógica de modelo de custo econômico 4996 estejam em uma margem aceitável de variância.

[00412] Em uma implementação, o identificador de qualidade 5276 identifica numericamente o nível de qualidade contribuído por uma mudança de elemento de GUI (por exemplo, valores de 0 a 10), determinado pela lógica de modelo de custo econômico 4996, correspondente a um especificador de mudança de GAP sintético gerado 4984. Em uma outra implementação, o identificador de qualidade 5276 identifica o nível de qualidade determinado por um testador e recebido pela lógica de modelo de custo econômico 4996 com o especificador de mudança de GAP 184. Distinguindo o

identificador de qualidade 5276 do especificador de mudança de GAP 184 e/ou de especificadores de mudança de GAP sintéticos 4984 a partir da classificação de qualidade 5260 da regra de custo de mudança de elemento de GUI 5246, o identificador de qualidade 5276 representa uma análise que é externa ao depósito de modelos econômicos 176. A lógica de modelo de custo econômico 4996 pode analisar a classificação de qualidade 5260 e o identificador de qualidade 5276 para avaliação da acurácia dos custos de transformação de GUI 5248 obtidos pela aplicação da regra de custo de mudança de elemento de GUI 5246. Por exemplo, a lógica de modelo de custo econômico 4996 pode determinar que a classificação de qualidade 5260 e o identificador de qualidade 5276 correspondentes a uma mudança de elemento de GUI em particular estão em uma margem aceitável de variância, de modo que o custo de transformação de GUI 5248 não seja ajustado, como resultado. Em um outro exemplo, a lógica de modelo de custo econômico 4996 pode determinar que a classificação de qualidade 5260 e o identificador de qualidade 5276 correspondentes a uma mudança de elemento de GUI em particular estão fora de uma margem aceitável de variância, e os custos de transformação de GUI 5248 são ajustados para cima por um multiplicador. A margem de variância e o multiplicador, determinados pela análise da classificação de qualidade 5260 e do identificador de qualidade 5276 podem ser selecionáveis por usuário e/ou ajustáveis.

[00413] Em uma implementação, o motor de custo econômico 182 recebe um especificador de mudança de GAP 184 que inclui um valor de percentagem de mudança 5278, um script de teste atual 5000 e um modelo de árvore de GAP atual correspondente a uma versão de GAP atual 150 que o motor de custo econômico 182 usa para a geração de especificadores de mudança de GAP sintéticos 4984, e localização e recuperação de custos de transformação de GUI 5248 a partir do

depósito de modelos econômicos 176. Por exemplo, a lógica de modelo de custo econômico 4996 analisa a versão de GAP atual 150 (por exemplo, representada por um modelo de árvore de GAP atual) e gera os especificadores de mudança de GAP sintéticos 4984 que refletem uma percentagem de mudança da versão de GAP atual 150 correspondente ao valor de percentagem de mudança 5278 (por exemplo, variando de 1 a 100). A lógica de modelo de custo econômico 4996 analisa a versão de GAP atual 150 e identifica um conjunto de mudanças de elemento de GUI propostas que correspondem ao valor de percentagem de mudança 5278. A lógica de modelo de custo econômico 4996 pode identificar os elementos de GUI propostos pela análise dos elementos de GUI no modelo de árvore de GAP atual da versão de GAP atual 150 em uma ordem randômica, a ordem na qual os elementos de GUI são apresentados no modelo de árvore a partir do topo para o fundo ou do fundo para o topo.

[00414] Em uma implementação, as mudanças de elemento de GUI propostas podem ser determinadas com base no identificador de complexidade 5274 e/ou no identificador de qualidade 5276 incluídos no especificador de mudança de GAP recebido 184. Por exemplo, a lógica de modelo de custo econômico 4996 recebe um especificador de mudança de GAP 184 que inclui um valor de identificador de complexidade 5274 de 1 e um valor de identificador de qualidade 5276 de 2, e para cada um dos elementos de GUI propostos a serem mudados determina mudanças propostas correspondentes ao valor de identificador de complexidade 5274 de 1 e ao valor de identificador de qualidade 5276 de 2. A lógica de modelo de custo econômico 4996 pode localizar, no depósito de medidas de performance 4998 e/ou no depósito de relatórios de custo 5288, mudanças de elemento de GUI propostas correspondentes aos valores de identificador de

complexidade 5274 e aos valores de identificador de qualidade 5276. Em uma implementação, a lógica de modelo de custo econômico 4996 gera especificadores de mudança de GAP sintéticos 4984, como resultado da análise das mudanças de elemento de GUI propostas. Em uma outra implementação, a lógica de modelo de custo econômico 4996 identifica as mudanças de elemento de GUI propostas correspondentes ao identificador de complexidade 5274, ao identificador de qualidade 5276 e aos coeficientes de habilidades 5272.

[00415] A lógica de modelo de custo econômico 4996 analisa o script de teste atual 5000 e o modelo de diferença de GUI 162 para a geração de especificadores de mudança de GAP sintéticos 4984 com base em mudanças de elemento de GUI validadas (por exemplo, entradas de diferença de elemento de GUI). Por exemplo, a lógica de modelo de custo econômico 4996 determina vetores de declaração de script de teste 3916 que precisam de modificação porque os objetos de GUI que são referenciados no script de teste atual 5000 e existem na versão de GAP atual 150 não existem na versão de GAP subsequente 152, e a lógica de modelo de custo econômico 4996 gera especificadores de mudança de GAP sintéticos 4984 que refletem as mudanças necessárias para o script de teste atual 5000. A lógica de modelo de custo econômico 4996 identifica mudanças nos vetor de declaração de script de teste 3916 que regulam os valores de objetos de GUI que são compatíveis com a classe daqueles objetos de GUI, de modo que restrições impostas nos objetos de GUI como resultado de uma mudança não sejam violadas. Em uma implementação, a lógica de modelo de custo econômico 4996 verifica que operações incorretas não são especificadas pelos especificadores de mudança de GAP 184 e/ou pelos especificadores de mudança de GAP sintéticos 4984 usados para a obtenção de custos de transformação de GUI

5248.

[00416] A lógica de modelo de custo econômico 4996 pode inferir uma informação de classe de GUI referente a um objeto de GUI que esteja presente em um caminho de navegação de vetores de declaração de script de teste 3916, e cuja presença não é explicitamente definida. Por exemplo, quando o vetor de declaração de script de teste 3916 emprega expressões que identificam objetos de GUI cujos valores podem ser determinados apenas no tempo de execução, a lógica de consulta de OR 172 pode usar a lógica de travessia de caminho 3924 para identificar possíveis entradas de objeto de GUI correspondentes 3922 e entradas de diferença de elemento de GUI 3910 no depósito de objeto 174 e no modelo de diferença de GUI 162, respectivamente. A lógica de modelo de custo econômico 4996 então identifica as entradas de objeto de GUI válidas 3922 que podem substituir as expressões e as entradas de diferença de elemento de GUI 3910 que satisfazem a vetores de declaração de script de teste válidos 3916, e a lógica de modelo de custo econômico 4996 gera especificadores de mudança de GAP sintéticos 4984 correspondentes.

[00417] Por exemplo, considere o vetor de declaração de script de teste 3916: VBWindow("s").VBWindow(e1).VBWindow(e2).VBWindow("d"), onde o objeto de GUI de nó de origem é denominado "s", o objeto de GUI de nó de destino é denominado "d", mas as expressões e1 e e2 computam valores de nós intermediários no caminho de navegação no tempo de execução. A lógica de travessia 3924 determina nós intermediários (objetos de GUI) que podem ser incluídos nos caminhos de navegação identificados pelo nó de origem "s" e pelo nó de destino "d". A lógica de travessia de caminho 3924 analisa o modelo de diferença de GUI 162 para identificar possíveis substituições de

constante por e1 e e2, por exemplo, "a" e "f", de modo que o vetor de declaração de script de teste 3916 formado pelos objetos de GUI substitutos na expressão de caminho de navegação "s.a.f.d" possa ser validado pela lógica de modelo de custo econômico 4996. Pela identificação dos possíveis caminhos de navegação levando ao nó de destino d começando com o nó de origem 's', a lógica de modelo de custo econômico 4996 pode concluir se é para gerar um especificador de mudança de GAP sintético 4984 com base nos objetos de GUI substitutos. No caso de a lógica de travessia 3924 não identificar pelo menos um caminho de navegação, então, a declaração de script de teste transformado 3928 será inválida. Alternativamente, no caso de a lógica de travessia 3924 identificar caminhos de navegação levando a partir de 's' para 'd' pela travessia de dois objetos (por exemplo, e1 e e2), então, a declaração de script de teste transformado 3928 poderá ser válida, desde que as expressões e1 e e2 avaliem para os nomes dos nós nos caminhos de navegação descobertos. A lógica de travessia 3924 infere os nomes possíveis computados pelas expressões e1 e e2 no tempo de compilação. Em outras palavras, há uma correlação direta entre a complexidade de scripts de teste e o custo econômico para a transformação de scripts de teste e o uso daqueles scripts de teste para se testarem versões de GAP subsequente 152. A complexidade é uma função do número de objetos de GUI referenciados e das operações nos objetos de GUI, bem como da quantidade de lógica necessária para processamento dos dados que são extraídos e colocados naqueles objetos de GUI.

[00418] Com referência à Figura 47, a lógica de modelo de custo econômico 4996 pode inferir uma informação de classe de GUI referente a objetos de GUI que estejam presentes no caminho de navegação de vetores de declaração de script de teste 3916. A lógica de modelo de custo econômico 4996 identifica os especificadores de

mudança de GAP 184 e/ou os especificadores de mudança de GAP sintéticos 4984 que resolvem vetores de declaração de script de teste 3916 que tentam acessar objetos de GUI que não existem em uma versão de GAP subsequente 152 e/ou tentam regular um valor de um objeto de GUI que não é compatível com o tipo do objeto de GUI. O tipo de lógica de modelo de custo econômico 4996 checa vetores de declaração de script de teste 3916 em relação ao modelo de diferença de GUI 162, antes da geração de especificadores de mudança de GAP 184 correspondentes.

[00419] A lógica de modelo de custo econômico 4996 usa relações de herança e subtipificação entre as classes de objetos de GUI para validação de especificadores de mudança de GAP recebidos 184 e geração de especificadores de mudança de GAP sintéticos válidos 4984. O conceito de classe inclui contenções hierárquicas (por exemplo, escopos de GUI e hierarquias de sistema). O depósito de objeto 174 e o modelo de diferença de GUI 162 incluem uma informação de classe de GUI (por exemplo, anotação das classes de objetos de GUI) para cada entrada de objeto de GUI 3922 e entrada de diferença de elemento de GUI 3910. Por exemplo, com referência à linha 1 da Tabela 12, a SchoolListBox é uma classe de WinObject com propriedades listadas nas linhas 3-39. Em um outro exemplo, com referência às Figuras 36, 45 e 46, na linha 1 de cada entrada de diferença de GUI (por exemplo, 3604, 4504 e 4604) o GUIElement Type é indicado. A classe de cada objeto de GUI é indicada conforme mostrado nas Figuras 36, 45 e 46, nas linhas 7, 8 e 8, respectivamente. A classe de um objeto de GUI indica que o objeto de GUI inclui atributos, propriedades e/ou tratos em particular em comum com outros objetos de GUI da mesma classe que podem ser estendidos e/ou herdados pelos objetos de GUI filhos. Por exemplo, a Figura 36 na linha 7 indica que o objeto de GUI StateListbox é de uma

classe WindowsForms10.ListBox.app4 que inclui valores, conforme indicado na linha 11 da Figura 36.

[00420] Com referência de novo à Figura 52, em uma implementação, a lógica de modelo de custo econômico 4996 determina se o objeto de GUI mudou e regula o status de mudança de elemento de GUI 3934. Por exemplo, o status de mudança de elemento de GUI 3934 pode usar um indicador numérico de 0, 1 e 2, respectivamente, para indicar nenhuma mudança e uma mudança com e sem uma violação de restrição em particular. A lógica de modelo de custo econômico 4996 pode usar o status de mudança de elemento de GUI 3934 para facilitar a identificação dos especificadores de mudança de GAP 184 e/ou especificadores de mudança de GAP sintéticos 4984 apropriados.

[00421] Em uma outra implementação, o status de mudança de elemento de GUI 3934 é uma mensagem que provê uma descrição detalhada de uma mudança. O status de mudança de elemento de GUI 3934 também pode indicar com um indicador numérico (por exemplo, -1) que o objeto de GUI foi apagado da versão de GAP subsequente 152. Quando um objeto de GUI foi apagado da versão de GAP subsequente 152, a lógica de modelo de custo econômico 4996 gera um ou mais especificadores de mudança de GAP sintéticos 4984 que especificam mudanças correspondentes no script de teste atual 5000 e na versão de GAP atual 150. Em uma implementação, a lógica de modelo de custo econômico 4996 gera especificadores de mudança de GAP sintéticos 4984 que correspondem a abordagens diferentes, mas equivalentes em termos programáticos, para mudança do script de teste atual 5000 e da versão de GAP atual 150, de modo que um programador possa avaliar os custos de transformação de GUI 5248 e um relatório de custo de transformação de script de teste.

[00422] A Figura 40 mostra um fluxograma 4000 para a

recuperação das propriedades de uma entrada de objeto de GUI 3922 a partir de um depósito de objeto (OR) 174. A lógica de analisador gramatical de script 3912 analisa gramaticalmente o vetor de declaração de script de teste 3916 em uma sequência ordenada de nós que representam funções e argumentos que navegam para objetos de GUI (4002). A lógica de analisador gramatical de script 3912 avalia o primeiro nó da sequência ordenada de nós (4004) e identifica o primeiro nó como o nó de origem e atribui um identificador de sequência indicando a posição do nó de origem na sequência ordenada (4006). A lógica de analisador gramatical de script 3912 avalia o próximo nó da sequência ordenada de nós para determinar se o próximo nó é o último nó (4008), e identifica o próximo nó como um nó intermediário, quando o próximo nó não for o último nó (4010). Ao nó intermediário é atribuído um identificador de sequência indicando a posição do nó intermediário na sequência ordenada. A lógica de analisador gramatical de script 3912 pode identificar todos os nós intermediários entre o nó de origem e o nó de destino.

[00423] A lógica de analisador gramatical de script 3912 identifica o último nó na sequência ordenada como o nó de destino e atribui um identificador de sequência ao nó de destino que indica a posição do nó de destino na sequência ordenada (4012). A OR Lookup 172 realiza uma consulta de depósito de objeto para cada objeto de GUI correspondente à sequência ordenada de nós para os quais o vetor de declaração de script de teste navega, de modo que cada entrada de objeto de GUI 3922 seja identificada (4014). Em uma implementação, a sequência ordenada de nós é usada pela lógica de travessia de caminho 3924 e pela lógica de modelo de custo econômico 4996 para validação das declarações do script de teste atual 5000 e/ou para validação dos especificadores de mudança de GAP recebidos 184 e dos especificadores de mudança de GAP sintéticos gerados 4984. Em

uma implementação, o motor de custo econômico 182 usa a sequência ordenada de nós para inferir a classe de GUI e uma informação de herança (subclasse) para objetos de GUI. Quando pelo menos um dentre os nós de origem, de destino e/ou intermediário é de expressões que podem ser identificadas apenas no tempo de execução, a lógica de travessia de caminho pode identificar possíveis entradas de objeto de GUI 3922, e a lógica de modelo de custo econômico 4996 e determina as entradas de objeto de GUI 3922 que satisfazem ao vetor de declaração de script de teste 3916. A OR Lookup 172 recupera as propriedades das entradas de objeto de GUI 3922 para as quais o vetor de declaração de script de teste navega (4016).

[00424] A Figura 53 mostra um fluxograma 5300 para a identificação de uma entrada de diferença de elemento de GUI 3910 correspondente a uma entrada de objeto de GUI 3922. A lógica de consulta de OR 172 recebe as propriedades dos objetos de GUI correspondentes a nós de origem, de destino e todos os intermediários do vetor de declaração de script de teste 3916. Em uma implementação, a lógica de consulta de OR 172 emprega a lógica de travessia de caminho 3924 para a identificação de possíveis entradas de diferença de elemento de GUI 3910 correspondentes aos caminhos de navegação identificados por um nó de origem e um nó de destino para o qual um vetor de declaração de script de teste navega (5302). Quando pelo menos uma das entradas de diferença de elemento de GUI 3910 é uma expressão que pode ser identificada apenas no tempo de execução, a lógica de travessia de caminho 3924 identifica uma ou mais possíveis entradas de diferença de elemento de GUI 3910 que formam um caminho de navegação entre o nó de origem e o nó de destino (5304). A lógica de travessia de caminho 3924 determina se as entradas de diferença de elemento de GUI 3910

formam um caminho de navegação válido entre as entradas de diferença de elemento de GUI 3910 de nós de origem e de destino correspondentes (5306). A lógica de modelo de custo econômico de GUI 4996 determina se as entradas de diferença de elemento de GUI 3910 que formam o caminho de navegação são válidas (5308).

[00425] A lógica de modelo de custo econômico 4996 identifica as entradas de diferença de elemento de GUI 3910 que correspondem a cada uma das entradas de objeto de GUI 3922 formando um caminho de navegação válido (5310). A lógica de modelo de custo econômico 4996 determina os especificadores de mudança de GAP sintéticos 4984 para a geração e/ou a validação dos especificadores de mudança de GAP 184 com base no tipo de mudança de elemento de GUI (por exemplo, 5280, 5282, 5284 e/ou o status de mudança de elemento de GUI 3934), com base na análise da entrada de objeto de GUI 3922 e da entrada de diferença de elemento de GUI 3910 (5312). A lógica de modelo de custo econômico 4996 gera especificadores de mudança de GAP sintéticos válidos 4984 correspondentes ao tipo de mudança de elemento de GUI identificado. Quando a lógica de travessia de caminho 3924 identifica um caminho de navegação que atravessa um número inválido de entradas de diferença de elemento de GUI 3910 entre as entradas de diferença de elemento de GUI 3910 de nó de origem e de destino correspondentes, a lógica de travessia de caminho 3924 indica que o caminho de navegação é inválido (5314).

[00426] A Figura 54 mostra um script de teste transformado 5400 para a versão de GAP subsequente 152. A lógica de modelo de custo econômico 4996 pode gerar especificadores de mudança de GAP sintéticos 4984 que especificam as mudanças necessárias para a obtenção do script de teste transformado 5400. Por exemplo, a lógica de modelo de custo econômico 4996 gera especificadores de

mudança de GAP sintéticos 4984 para as linhas 1-3 do script de teste atual 5000, mostrado na Figura 50, correspondente às linhas de script de teste transformado 1, 5-7, mostradas na Figura 54. Em uma implementação, o modelo de diferença de GUI 162 e os metadados de elemento de GUI provêm a classe de GUI, a tipificação de GUI e a informação de mapeamento necessárias para que a lógica de modelo de custo econômico 4996 infira as linhas 2-4 do script de teste transformado 5400, dado que o "university.data" na linha 6 representa um destino em uma travessia de caminho a partir da qual os especificadores de mudança de GAP válidos 184 e/ou os especificadores de mudança de GAP sintéticos 4984 podem ser determinados. Em um outro exemplo, o modelo de diferença de GUI 162 e/ou os metadados de elemento de GUI incluem uma informação de classe de GUI e de mapeamento que o motor de custo econômico 182 usa para gerar um ou mais especificadores de mudança de GAP sintéticos 4984 que especificam como transformar a linha 16 do script de teste atual 5000, conforme mostrado na Figura 50, que refere-se ao WinObject "Save File" nas linhas 24-25 que referem-se a um objeto de GUI filho "Save File" do WinObject "menuStrip1".

[00427] A Figura 55 mostra um fluxograma 5500 para a geração de um especificador de mudança de GAP sintético 4984. Em uma implementação, a lógica de modelo de custo econômico 4996 valida os especificadores de mudança de GAP 184 e/ou gera especificadores de mudança de GAP sintéticos 4984 que especificam o tipo de mudanças de objeto de GUI entre liberações de edições sucessivas de GAPs, incluindo: (A) um novo objeto de GUI adicionado a uma versão de GAP subsequente 152; (B) um objeto de GUI é apagado de uma versão de GAP subsequente 152; (C) os valores de um ou mais atributos de um objeto de GUI são modificados; (D) os valores de um objeto de GUI são modificados entre versões de GAP sucessivas; e

(E) o tipo de um objeto de GUI é diferente. O motor de custo econômico 182 analisa os objetos de GUI referidos no script de teste atual 5000, a versão de GAP atual 150 e os especificadores de mudança de GAP 184 (5502). O motor de custo econômico 182 recupera as propriedades de cada objeto de GUI (por exemplo, as entradas de objeto de GUI 3922) a partir do depósito de objeto 174, e localiza uma entrada de diferença de elemento de GUI 3910 correspondente no modelo de diferença de GUI 162 (5504). Em uma implementação, o motor de custo econômico 182 recebe uma representação de modelo de árvore de GAP atual de uma versão de GAP atual 150 a partir do modelo de diferença de GUI 162 e especificadores de mudança de GAP 184, e gera especificadores de mudança de GAP sintéticos 4984, usando a lógica de especificador de mudança de GAP 5290. A lógica de modelo de custo econômico 4996 analisa as mudanças de objeto de GUI (5506).

[00428] As mudanças de objeto de GUI de tipos A (5508) e B (5510) ocorrem quando objetos de GUI são adicionados e removidos de forma correspondente da versão de GAP atual 150 e da versão de GAP subsequente 152. Por exemplo, a adição do WinObject menustrip1 308 para a versão de GAP subsequente 152 é uma mudança de tipo A, enquanto a remoção do WinObject "Select School" 3304 é uma mudança de objeto de GUI de tipo B. Com referência à Figura 35, note que o "Select School" foi removido em 3512.

[00429] Um exemplo de uma mudança de tipo C (5512) é a mudança do nome de janela de StateList 3302 para School 3402. A adição ou remoção de valores de objetos de GUI, tais como caixas de lista e de combinação, é um exemplo de modificações da mudança de tipo D (5512). Por exemplo, a caixa de lista StateListbox 3312 na versão de GAP atual 150 é identificada na versão de GAP subsequente 152 como StateListbox 3404, e com referência à entrada

de diferença de GUI 3604 os valores para SeqNumber = "8" são "District of Columbia" e "Florida" para versões de GAP sucessivas, respectivamente. A mudança do "type" de um objeto de GUI pode incluir a substituição de uma classe da janela que é usada para representação do objeto e/ou a mudança do conceito de nível alto que descreve os valores que o objeto de GUI assume. Por exemplo, a mudança do tipo do rótulo estático para uma caixa de combinação apenas de leitura é uma modificação do tipo E (5512). Um outro exemplo de uma mudança de tipo E inclui a mudança da caixa de lista "SchooListbox" 3316 para uma caixa de combinação "SchoolCombobox" 3406.

[00430] A lógica de modelo de custo econômico 4996 recebe os especificadores de mudança de GAP 184 e gera especificadores de mudança de GAP sintéticos 4984 que incluem os especificadores de tipo de apagamento de percurso errado 5280, de tipo de mesmo percurso errado 5282 e de tipo de elemento mudado 5284. O tipo de apagamento de percurso errado 5280 especifica que um objeto de GUI na versão de GAP atual 150 pode ter sido apagado na versão de GAP subsequente 152 (por exemplo, vide "Select School" 3318 e 3512 conforme mostrado na Figura 35), embora o script de teste atual 5000 se refira ao objeto de GUI (5514). O tipo de mesmo percurso errado 5282 especifica que uma mudança de GAP pode resultar em uma leitura e/ou uma escrita para o objeto de GUI errado. Por exemplo, o método pode ser invocado no objeto de GUI errado com base em uma mudança de GAP em particular. O tipo de mesmo percurso errado 5282 especifica que um objeto de GUI em uma versão de GAP atual 150 foi modificado e/ou um outro objeto de GUI foi adicionado à versão de GAP subsequente 152, que pode resultar em o objeto de GUI ser navegado para por uma declaração de script de teste (5516).

[00431] Por exemplo, considere a declaração nas linhas 2 e 6 do

script de teste atual 5000 e do script de teste transformado 5400, respectivamente:

[00432] Window("StateList").Dialog("Open").WinListView("SysListView32").Select "university.data".

[00433] A declaração seleciona "university.data" a partir de uma WinListView "SysListView32". Contudo, as linhas 2-4 do script de teste transformado 5400 podem navegar para e invocar o método Select no objeto de GUI errado "university.data", porque os objetos de GUI referenciados nas linhas 204 do script de teste transformado 5400 são objetos de GUI novos que não são referenciados no script de teste atual 5000. Assim, quando as propriedades de objetos de GUI existentes são modificadas e/ou outros objetos de GUI são adicionados em uma versão de GAP subsequente 152, o resultado de interferência destas operações é que as declarações de script de teste transformado que resultam da aplicação de especificadores de mudança de GAP 184 e/ou especificadores de mudança de GAP sintéticos 4984 em vetores de declaração de script de teste atuais 3916 podem acessar e ler valores de objetos que são diferentes daqueles originalmente pretendidos.

[00434] O elemento mudado 5284 especifica que o tipo, as propriedades e/ou os valores-padrão de um objeto de GUI referenciados por um vetor de declaração de script de teste 3916 mudaram na versão de GAP subsequente 152 (5518). Por exemplo, a entrada de diferença de GUI 3604 indica que há valores diferentes para SeqNumber = "8" são "District of Columbia" e "Florida" para versões de GAP sucessivas, e a lógica de modelo de custo econômico 4996 pode gerar um especificador de mudança de GAP sintético 4984 que inclui um elemento de mudança 5284 de forma correspondente. O elemento de mudança 5284 também pode especificar que novas restrições foram impostas em um objeto de GUI que entram em

conflito com os vetores de declaração de script de teste 3916, por exemplo, tentando escrever dados em uma caixa de texto que podia ser escrita previamente que mudou para uma caixa de texto apenas de leitura.

[00435] Com referência à entrada de diferença de elemento de GUI mostrada na Tabela 13, o WinObject "AcadScale" referido no script de teste atual 5000 na linha 8 é um objeto editável que foi transformado no WinObject "Academics (1-5)" na versão de GAP subsequente 152 em que o objeto é apenas de leitura. A lógica de modelo de custo econômico 4996 valida os especificadores de mudança de GAP 184 e/ou gera o especificador de mudança de GAP sintético 4984 com o tipo de mudança de GAP especificado (5520), e o status de mudança de elemento de GUI 3934 é atualizado (5522). Em uma implementação, a lógica de modelo de custo econômico 4996 não gera especificadores de mudança de GAP sintéticos 4984 para objetos de GUI que não tenham mudado entre versões de GAP sucessivas (5524).

[00436] Conhecer o tipo de modificação para um objeto de GUI facilita a lógica de modelo de custo econômico 4996 na determinação dos especificadores de mudança de GAP sintéticos apropriados 4984 para a geração e/ou validação de especificadores de mudança de GAP recebidos 184. Por exemplo, a lógica de modelo de custo econômico 4996 pode validar os especificadores de mudança de GAP 184 e/ou gerar um ou mais especificadores de mudança de GAP sintéticos 4984 que especificam mudanças em vetores de declaração de script de teste 3916 que tentam regular valores em um objeto de caixa de texto que mudou para uma caixa de combinação de apenas leitura. Os especificadores de mudança de GAP 184 e/ou os especificadores de mudança de GAP sintéticos 4984 podem especificar que o vetor de declaração de script de teste 3916 foi

modificado (transformado) para a seleção de valores na caixa de combinação usando-se interfaces apropriadas, ao invés de se tentar regular valores na caixa de texto.

[00437] A lógica de modelo de custo econômico 4996 determina se os objetos de GUI foram removidos de uma versão de GAP atual 150 e localiza os vetores de declaração de script de teste 3916 que referenciam estes objetos removidos no script de teste atual 5000. O motor de custo econômico 182 refere-se a estas declarações como declarações de primeira referência (FRS). As variáveis usadas nestas declarações são obtidas e as declarações que usam as variáveis cujos valores são definidos nas FRSs são referidas como declarações de referência secundária (SRS). A lógica de modelo de custo econômico 4996 determina se os objetos de GUI podem ter sido apagados na versão de GAP subsequente 152, e valida os especificadores de mudança de GAP recebidos 184 e gera um ou mais especificadores de mudança de GAP sintéticos 4984 correspondentes com um apagamento de caminho errado 5284. Quando uma declaração do script de teste atual 5000 refere-se a uma variável cujo valor aponta para um objeto de GUI removido, a declaração do script de teste transformado 3926 é considerada um SRS. Em uma implementação, o motor de custo econômico 182 gera um ou mais especificadores de mudança de GAP sintéticos 4984 e/ou valida os especificadores de mudança de GAP recebidos 184 correspondentes as SRSs identificadas.

[00438] Quando os valores de um ou mais atributos de um objeto de GUI são modificados, uma modificação de tipo C é realizada. As FRSs e SRSs são identificadas para o objeto de GUI com os atributos modificados e especificadores de mudança de GAP sintéticos 4984 correspondentes são gerados e/ou os especificadores de mudança de GAP recebidos 184 são validados. Quando os valores de objetos de

GUI são adicionados ou removidos, as modificações do tipo D ocorrem. Após a localização de FRSs que referenciam objetos de GUI cujos valores foram mudados, SRSs são encontradas e o motor econômico 182 determina o impacto devido as SRSs. Quando o tipo de um objeto de GUI é modificado, então, uma modificação do tipo E ocorre, que envolve a localização de FRSs, a checagem de novos tipos de objeto de GUI, invocando regras de subsunção de tipo correspondente. A lógica de modelo de custo econômico 4996 pode analisar os objetos de GUI modificados para determinar se é para gerar especificadores de mudança de GAP sintéticos 4984 com o tipo de elemento mudado 5284, onde objetos de GUI cujos tipos, propriedades ou valores-padrão são mudados em uma versão de GAP subsequente 152, e/ou tentando uma operação em um objeto de GUI que não leva em consideração novas restrições impostas nos elementos do objeto de GUI.

[00439] A lógica de modelo de custo econômico 4996 analisa cada objeto de GUI referido no script de teste atual 164 e/ou no script de teste atual 5000, a versão de GAP atual 150, os especificadores de mudança de GAP recebidos 184, os especificadores de mudança de GAP sintéticos gerados 4984 e/ou o status de mudança de elemento de GUI 3934. A lógica de modelo de custo econômico 4996 localiza, no depósito de modelos econômicos 176, o modelo econômico especificado pelo especificador de modelo 5268, e recupera uma regra de custo de mudança de elemento de GUI 5246 com um identificador de especificador de mudança 5250 correspondente ao especificador de mudança de GAP 184 e/ou ao especificador de mudança de GAP sintético 4984. Em uma implementação, a lógica de modelo de custo econômico 4996 combina um ou mais atributos de um objeto de GUI (por exemplo, tipo e/ou classe) com o status de mudança de elemento de GUI 3934, o especificador de modelo 5268, o tipo de apagamento

de percurso errado 5280, o tipo de mesmo percurso errado 5282 e/ou o tipo de elemento mudado 5284 para a formação de um identificador único usado para a localização de um identificador de especificador de mudança 5250 correspondente no modelo econômico especificado pelo especificador de modelo 5268.

[00440] A lógica de modelo de custo econômico 4996 analisa os componentes de regra de custo de mudança de elemento de GUI 5246, o especificador de mudança de GAP 184 e/ou os componentes de especificador de mudança de GAP sintético 4984, os identificadores de preferência 5266 e o fator de eficiência de mudança de GUI 5286 para se determinar se é para ajustar a estimativa de custo de mudança de GUI 5254. Por exemplo, a lógica de modelo de custo econômico 4996 ajusta a estimativa de custo de mudança de GUI 5254 com base em se os coeficientes de habilidades 5272, identificador de complexidade 5274, identificador de qualidade 5276, identificadores de utilização de recurso de sistema 5252, classificação de qualidade 5260 e/ou classificação de complexidade 5262 estão em uma variância aceitável, conforme especificado pelos identificadores de preferência 5266. A lógica de modelo de custo econômico 4996 obtém o custo de transformação de GUI 5248 com base na estimativa de custo de mudança de GUI 5254. Em outras palavras, a estimativa de custo de mudança de GUI 5254 é ajustada para a obtenção do custo de transformação de GUI para o especificador de mudança de GAP 184 e/ou o especificador de mudança de GAP sintético 4984. A lógica de modelo de custo econômico 4996 processa cada especificador de mudança de GAP recebido 184 e/ou especificador de mudança de GAP sintético gerado 4984 para a obtenção dos custos de transformação de GUI 5248 correspondentes e gera o relatório de custo de transformação de script de teste 186 com os custos de transformação de GUI 5248.

[00441] A Figura 56 mostra um fluxograma para extração de um relatório de custo de transformação de script de teste 186 com base em um modelo de diferença de GUI 162. O motor de custo econômico 182 recebe o modelo de diferença de GUI 162 e os metadados de elemento de GUI 140 (5604). O motor de custo econômico 182 recebe uma representação de script de teste atual 3914 que inclui um vetor de declaração de script de teste 3916 (5606). A lógica de analisador gramatical de script 3912 analisa gramaticalmente o vetor de declaração de script de teste 3916 em nós de vetor para determinar os objetos de GUI para os quais o vetor de declaração de script de teste 3916 navega (5608). O motor de custo econômico 182 invoca a lógica de consulta de OR 172 para cada objeto de GUI identificado pelo vetor de declaração de script de teste 3916 para a recuperação das propriedades dos objetos de GUI a partir do depósito de objeto 174 (5610). A lógica de travessia de caminho 3924 analisa o caminho de navegação das entradas de diferença de elemento de GUI 3910 que correspondem aos objetos de GUI identificados pelo vetor de declaração de script de teste 3916 (5612). A lógica de modelo de custo econômico 4996 valida um especificador de mudança de GAP 184 e/ou determina o tipo de especificador de mudança de GAP sintético 4984 a gerar (5614). A lógica de modelo de custo econômico 4996 gera um especificador de mudança de GAP sintético 4984 correspondente ao tipo de mudança de elemento de GUI identificado pela análise do script de teste atual 164, da versão de GAP atual 150 (por exemplo, o modelo de árvore de GAP atual) e o modelo de diferença de GUI 162 (5616). A lógica de modelo de custo econômico 4996 localiza no modelo econômico especificado pelo especificador de modelo 5268 a regra de custo de mudança de elemento de GUI 5246 correspondente ao especificador de mudança de GAP 184 e/ou ao especificador de mudança de GAP sintético 4984 e aplica a regra de

custo de mudança de elemento de GUI 5246 para a obtenção do custo de transformação de GUI 5248 (5618). A lógica de modelo de custo econômico 4996 gera o relatório de custo de transformação de script de teste 186 com base no custo de transformação de GUI 5248 (5620).

[00442] Em uma implementação, a arquitetura de motor de custo econômico 110 usa uma programação adaptativa incluindo classe e gráficos de objeto e uma abstração que trata todos os objetos uniformemente. A lógica de travessia de caminho 3924 e a lógica de modelo de custo econômico 4996 podem distinguir tipos complexos e simples de objetos de GUI. Dado um objeto de GUI de algum tipo, a lógica de travessia 3924 e a lógica de modelo de custo econômico 4996 trabalham em conjunto para a identificação de um ou mais objetos alcançáveis que satisfazem a certos critérios. A tarefa realizada é equivalente a determinar se os vetores de declaração de script de teste 3916 que descrevem os caminhos de navegação são válidos. A tarefa de checagem de scripts de teste (por exemplo, scripts de teste transformados 5400) é grandemente simplificada quando os nomes de nomes de componentes estranhos são definidos como constantes de string. Quando os nomes de objetos de GUI são especificados usando expressões, os valores destas expressões não podem ser determinados usando-se expressões, os valores destas expressões podem não ser determinados até um tempo de execução. Os gráficos de tipo facilitam o sistema de motor econômico 5200 para inferir tipos de expressões e variáveis que mantêm os nomes de objetos de GUI. O sistema de motor econômico 5200 aplica conceitos com base na Análise de Gráfico de Travessia (TGA) definida em uma programação adaptativa para se inferirem tipos de expressões e variáveis.

[00443] Quando os valores de expressões de string nas

declarações de script de teste não podem ser computados até o tempo de execução, as expressões de string podem ser inferidas. A lógica de travessia de caminho 3924 e a lógica de modelo de custo econômico 4996 trabalham em conjunto para a análise de vetores de declaração de script de teste 3916, usando-se gráficos de tipo pela transformação de vetores de declaração de script de teste 3916 em uma estratégia adaptativa com variáveis substituindo expressões de string. A lógica de modelo de custo econômico 4996 computa valores possíveis para cada variável e gera caminhos de travessia para cada estratégia. Quando pelo menos um caminho é identificado, então, um especificador de mudança de GAP 184 correspondente é validado e/ou um especificador de mudança de GAP sintético 4984 é gerado, uma vez que valores de expressões que computam nomes de objetos podem não estar nos caminhos computados. A lógica de travessia de caminho 3924 identifica um ou mais caminhos possíveis, enquanto a lógica de modelo de custo econômico 4996 valida caminhos para as expressões e as declarações.

[00444] O sistema de motor econômico 5200 provê uma integridade de modularização como um mecanismo para garantia da validade dos especificadores de mudança de GAP 184 e/ou especificadores de mudança de GAP sintéticos 4984. A integridade de modularização especifica que cada declaração de script de teste atual identificada por um especificador de mudança de GAP 184 e/ou um especificador de mudança de GAP sintético 4984 a ser mudada possa se comunicar diretamente apenas com objetos que pertençam a GUIs para as quais a declaração de script de teste atual, conforme mudada pelo especificador de mudança de GAP 184 e/ou por um especificador de mudança de GAP sintético 4984, é criada. As composições de declarações de script de teste atuais mudadas conforme especificado pelos especificadores de mudança de GAP 184 e/ou especificadores

de mudança de GAP sintéticos 4984, nas quais os objetos de GUI são acessados ao se chamarem funções exportadas pelas declarações de script de teste atuais mudadas conforme especificado, não devem violar a integridade de modularização. O sistema de motor econômico 5200 assegura a integridade de modularização de especificadores de mudança de GAP 184 e/ou especificadores de mudança de GAP sintéticos 4984 pela análise de composições de declarações de script de teste atuais mudadas conforme especificado por especificadores de mudança de GAP 184 e/ou especificadores de mudança de GAP sintéticos 4984 para a construção de relações transitivas entre o script de teste atual 164 e o script de teste atual 164 mudado conforme especificado pelos especificadores de mudança de GAP 184 e/ou especificadores de mudança de GAP sintéticos 4984 (por exemplo, o script de teste transformado 178 e o 5400).

[00445] Por exemplo, uma declaração Func("y", "z"), encontrada em uma suíte de scripts de teste relacionados, navega para o campo z de objeto de GUI estranho y em alguns scripts de teste que exportam a função Func. Assim, alguns scripts de teste na suíte de scripts de teste relacionados podem violar a integridade de modularização ao interoperarem de forma implícita os scripts de teste através da função Func, embora esta comunicação possa ser proibida pelas restrições de uma dada suíte de teste. Em uma implementação, o sistema de motor econômico 5200 codifica as restrições de modularização quando definindo scripts de teste usando as restrições de palavra chave como parte de um comentário global em cada script de teste. Estas restrições definem GAPs e suas telas de GUI, bem como outros scripts de teste com os quais um dado script de teste pode se comunicar. Um exemplo é uma declaração que especifica uma restrição que é 'constraints screen("Q") test_scripts("P, S"). Esta restrição efetivamente proíbe um dado script de teste de se comunicar com

outros GAPs, telas de GUI e scripts de teste, exceto a tela Q e os scripts de teste P e S, de forma explícita ou implícita.

[00446] A complexidade de tempo da lógica de travessia de caminho 3924 e da lógica de modelo de custo econômico 4996 é exponencial para o tamanho do gráfico de tipo para cada script de teste 164. Devido ao fato de a lógica de travessia de caminho 3924 e a lógica de modelo de custo econômico 4996 envolverem a busca de um ou mais nós e bordas no gráfico de tipo que contém ciclos para cada nó na estratégia, a complexidade de tempo é $O((V+ E)^{max(|\pi|)})$, onde V é o número de nós, E é o número de bordas no gráfico de tipo e max(|π|) é o número máximo de nós em estratégias. As operações de armazenamento de sucessores na tabela de variáveis assumem $O(1)$. Em geral, o número de nós max(|π|) em estratégias é muito menor do que o número de nós em gráficos de tipo. Nem todos os nós de gráfico podem precisar ser explorados para cada nó em uma estratégia.

[00447] Os sistemas podem ser implementados de muitas formas diferentes. Por exemplo, embora alguns recursos sejam mostrados armazenados em memórias que podem ser lidas em computador (por exemplo, como uma lógica implementada como instruções executáveis em computador ou como estruturas de dados em memória), todos ou parte dos sistemas, da lógica e das estruturas de dados podem ser armazenados em, distribuídos através de ou lidos a partir de outros meios que podem ser lidos em máquina. Os meios podem incluir discos rígidos, discos flexíveis, CD-ROMs, um sinal, tal como um sinal recebido a partir de uma rede ou dividido em seções e recebido em múltiplos pacotes comunicados através de uma rede. Os sistemas podem ser implementados em software, hardware ou uma combinação de software e de hardware.

[00448] Além disso, os sistemas podem ser implementados com componentes adicionais, diferentes ou menos. Como um exemplo, um

processador ou qualquer outra lógica pode ser implementado com um microprocessador, um microcontrolador, um DSP, um circuito integrado específico de aplicativo (ASIC), instruções de programa, lógica analógica ou digital discreta, ou uma combinação de outros tipos de circuitos ou lógica. Como um outro exemplo, memórias podem ser DRAM, SRAM, flash ou outro tipo de memória. Os sistemas podem ser distribuídos dentre múltiplos componentes, tal como dentro múltiplos processadores e memórias, opcionalmente incluindo múltiplos sistemas de processamento distribuídos. Uma lógica, tais como programas ou circuitos, pode ser combinada ou dividida dentre múltiplos programas, distribuída através de várias memórias e processadores, e pode ser implementada em ou como uma biblioteca de função, tal como uma biblioteca de enlace dinâmico (DLL) ou outra biblioteca compartilhada.

[00449] Embora várias modalidades tenham sido descritas, será evidente aproximadamente aqueles de conhecimento comum na técnica que muitas modalidades a mais e implementações são possíveis no escopo da invenção. Assim sendo, a invenção não é para ser restrita, exceto à luz das reivindicações anexadas e seus equivalentes.

REIVINDICAÇÕES

1. Produto para análise de transformação de script de teste para gerar um script de teste transformado para uma versão de aplicativo de interface gráfica de usuário (GAP) subsequente surgindo a partir de uma versão de GAP atual, para testar o GAP subsequente, o produto **caracterizado pelo fato de que** compreende:

um meio legível por computador não transitório;

uma lógica de modelo de diferença (2380, 3908) armazenada no meio legível por máquina operável para produzir um modelo de diferença de interface gráfica de usuário (GUI) capturando uma mudança de elemento de GUI a partir do GAP atual para o GAP subsequente para um elemento de GUI específico; e

lógica de análise de script (3920) armazenada no meio legível por máquina operável para:

receber o modelo de diferença de GUI;

receber uma representação de script de teste atual (3800, 3914, 5100) para um script de teste atual (164, 5000) para testar o GAP atual, incluindo um vetor de declaração de script de teste (3916);

localizar, em um depósito de objeto, uma entrada de objeto de GUI combinando com o vetor de declaração de script de teste (3916);

localizar, no modelo de diferença de GUI, uma entrada de diferença de elemento de GUI combinando com a entrada de objeto de GUI;

analisar a entrada de diferença de elemento de GUI para determinar se o elemento de GUI específico mudou; e

extrair um especificador de mudança (184, 5256) para o script de teste atual, quando a análise de elemento de GUI determinar que o elemento de GUI mudou, em que o especificador de mudança (184, 5256) identifica: uma mensagem de guia de mudança de tipo de

apagar caminho errado, uma mensagem de guia de mudança de tipo de mesmo caminho errado, ou uma mensagem de guia de mudança de tipo de elemento mudado, ou qualquer combinação dos mesmos.

2. Produto, de acordo com a reivindicação 1, **caracterizado pelo fato de que** a mensagem de guia de mudança de tipo de apagar caminho errado indica que um primeiro elemeto de GUI da versão de GAP atual é apagado da versão de GAP subsequente.

3. Produto, de acordo com a reivindicação 1, **caracterizado pelo fato de que** uma mensagem de guia de mudança do tipo de mesmo caminho errado indica que um primeiro elemento de GUI da versão de GAP atual e um segundo elemento de GUI da versão de GAP subsequente têm identificadores combinando, mas os primeiro e segundo elementos de GUI não representam o mesmo elemento de GUI.

4. Produto, de acordo com a reivindicação 1, **caracterizado pelo fato de que** a mensagem de guia de mudança de tipo de elemento mudado indica que um primeiro elemento de GUI da versão de GAP atual e um segundo elemento de GUI da versão de GAP subsequente têm identificadores combinando, mas pelo menos um atributo do primeiro elemento de GUI e do segundo elemento de GUI são diferentes.

5. Produto, de acordo com a reivindicação 1, **caracterizado pelo fato de que** a lógica de análise de script (3920) é ainda operável para receber regras de mudança de script de elemento de GUI e analisar a entrada de diferença de elemento de GUI com base nas regras de mudança de script de elemento de GUI para determinar se extrai o especificador de mudança (184, 5256) como uma declaração de script de teste transformado.

6. Produto, de acordo com a reivindicação 1, **caracterizado pelo fato de que** a lógica de análise de script (3920) é ainda operável

para receber regras de mudança de script de elemento de GUI e analisar a entrada de diferença de elemento de GUI com base nas regras de mudança de script de elemento de GUI para obter o especificador de mudança (184, 5256) como uma declaração de script de teste transformado.

7. Produto, de acordo com a reivindicação 6, **caracterizado pelo fato de que** a declaração de script de teste transformado facilita um teste de uma propriedade de exibição do GAP subsequente.

8. Produto, de acordo com a reivindicação 6, **caracterizado pelo fato de que** a declaração de script de teste transformado facilita um teste de uma operação de navegação do GAP subsequente.

9. Produto, de acordo com a reivindicação 6, **caracterizado pelo fato de que** a declaração de script de teste transformado facilita um teste de uma ação do GAP subsequente.

10. Sistema para análise de transformação de script de teste para gerar um script de teste transformado para uma versão de aplicativo de interface gráfica de usuário (GAP) subsequente surgindo a partir de uma versão de GAP atual para testar o GAP subsequente, o sistema **caracterizado pelo fato de que** compreende:

um processador;

uma memória não transitória acoplada ao processador, a memória compreendendo:

uma lógica de modelo de diferença (2380, 3908) operável para produzir um modelo de diferença de interface gráfica de usuário (GUI) capturando uma mudança de elemento de GUI a partir do GAP atual para o GAP subsequente para um elemento de GUI específico; e

lógica de análise de script (3920) operável para:

receber o modelo de diferença de GUI;

receber uma representação de script de teste atual (3800, 3914, 5100) para um script de teste atual (164, 5000) para testar o

GAP atual, incluindo um vetor de declaração de script de teste (3916);

localizar, em um depósito de objeto, uma entrada de objeto de GUI combinando com o vetor de declaração de script de teste (3916);

localizar, no modelo de diferença de GUI, uma entrada de diferença de elemento de GUI combinando com a entrada de objeto de GUI;

analisar a entrada de diferença de elemento de GUI para determinar se o elemento de GUI específico mudou; e

extrair um especificador de mudança (184, 5256) para o script de teste atual, quando a análise de elemento de GUI determina que o elemento de GUI mudou, em que o especificador de mudança (184, 5256) identifica: uma mensagem de guia de mudança de tipo de apagar caminho errado, uma mensagem de guia de mudança de tipo de mesmo caminho errado, ou uma mensagem de guia de mudança de tipo de elemento mudado, ou qualquer combinação dos mesmos.

11. Sistema, de acordo com a reivindicação 10, **caracterizado pelo fato de que** a mensagem de guia de mudança de tipo de apagar caminho errado indica que um primeiro elemeto de GUI da versão de GAP atual é apagado da versão de GAP subsequente. .

12. Sistema, de acordo com a reivindicação 10, **caracterizado pelo fato de que** a mensagem de guia de mudança de tipo de mesmo caminho errado indica que um primeiro elemento de GUI da versão de GAP atual e um segundo elemento de GUI da versão de GAP subsequente têm identificadores combinando, mas os primeiro e segundo elementos de GUI não representam o mesmo elemento de GUI.

13. Sistema, de acordo com a reivindicação 10, **caracterizado pelo fato de que** a mensagem de guia de mudança de tipo de elemento mudado indica que um primeiro elemento de GUI da

versão de GAP atual e um segundo elemento de GUI da versão de GAP subsequente têm identificadores combinando, mas pelo menos um atributo do primeiro elemento de GUI e do segundo elemento de GUI são diferentes.

14. Sistema, de acordo com a reivindicação 10, **caracterizado pelo fato de que** a lógica de análise de script (3920) é ainda operável para receber regras de mudança de script de elemento de GUI e analisar a entrada de diferença de elemento de GUI com base nas regras de mudança de script de elemento de GUI para determinar se extrai o especificador de mudança (184, 5256) como uma declaração de script de teste transformado.

15. Sistema, de acordo com a reivindicação 10, **caracterizado pelo fato de que** a lógica de análise de script (3920) é ainda operável para receber regras de mudança de script de elemento de GUI e analisar a entrada de diferença de elemento de GUI com base nas regras de mudança de script de elemento de GUI para obter o especificador de mudança (184, 5256) como uma declaração de script de teste transformado.

16. Método para testar análise de transformação de script de teste para gerar um script de teste transformado para uma versão de aplicativo de interface gráfica de usuário (GAP) subsequente surgindo a partir de uma versão de GAP atual, para testar o GAP subsequente, o método **caracterizado pelo fato de que** compreende:

produzir um modelo de diferença de interface gráfica de usuário (GUI) capturando uma mudança de elemento de GUI a partir do GAP atual para o GAP subsequente para um elemento de GUI específico;

receber o modelo de diferença de GUI;

receber uma representação de script de teste atual (3800, 3914, 5100) para um script de teste atual (164, 5000) para testar o

GAP atual, incluindo um vetor de declaração de script de teste (3916);

localizar, em um depósito de objeto, uma entrada de objeto de GUI combinando com o vetor de declaração de script de teste (3916);

localizar, no modelo de diferença de GUI, uma entrada de diferença de elemento de GUI combinando com a entrada de objeto de GUI;

analisar a entrada de diferença de elemento de GUI para determinar se o elemento de GUI específico mudou; e

extrair um especificador de mudança (184, 5256) para o script de teste atual, quando a análise de elemento de GUI determina que o elemento de GUI mudou, em que o especificador de mudança (184, 5256) identifica: uma mensagem de guia de mudança de tipo de apagar caminho errado, uma mensagem de guia de mudança de tipo de mesmo caminho errado, ou uma mensagem de guia de mudança de tipo de elemento mudado, ou qualquer combinação dos mesmos.

17. Método, de acordo com a reivindicação 16, **caracterizado pelo fato de que** a mensagem de guia de mudança de tipo de apagar caminho errado indica que um primeiro elemeto de GUI da versão de GAP atual é apagado da versão de GAP subsequente.

18. Método, de acordo com a reivindicação 16, **caracterizado pelo fato de que** a mensagem de guia de mudança de tipo de mesmo caminho errado indica que um primeiro elemento de GUI da versão de GAP atual e um segundo elemento de GUI da versão de GAP subsequente têm identificadores combinando, mas os primeiro e segundo elementos de GUI não representam o mesmo elemento de GUI.

19. Método, de acordo com a reivindicação 16, **caracterizado pelo fato de que** a mensagem de guia de mudança de tipo de elemento mudado indica que um primeiro elemento de GUI da

versão de GAP atual e um segundo elemento de GUI da versão de GAP subsequente têm identificadores combinando, mas pelo menos um atributo do primeiro elemento de GUI e do segundo elemento de GUI são diferentes.

20. Método, de acordo com a reivindicação 16, **caracterizado pelo fato de que** ainda compreende analisar a entrada de diferença de elemento de GUI com base nas regras de mudança de script de elemento de GUI para determinar se extrai o especificador de mudança (184, 5256) como uma declaração de script de teste transformado.

21. Método, de acordo com a reivindicação 16, **caracterizado pelo fato de que** ainda compreende analisar a entrada de diferença de elemento de GUI com base nas regras de mudança de script de elemento de GUI para obter o especificador de mudança (184, 5256) como uma declaração de script de teste transformado.

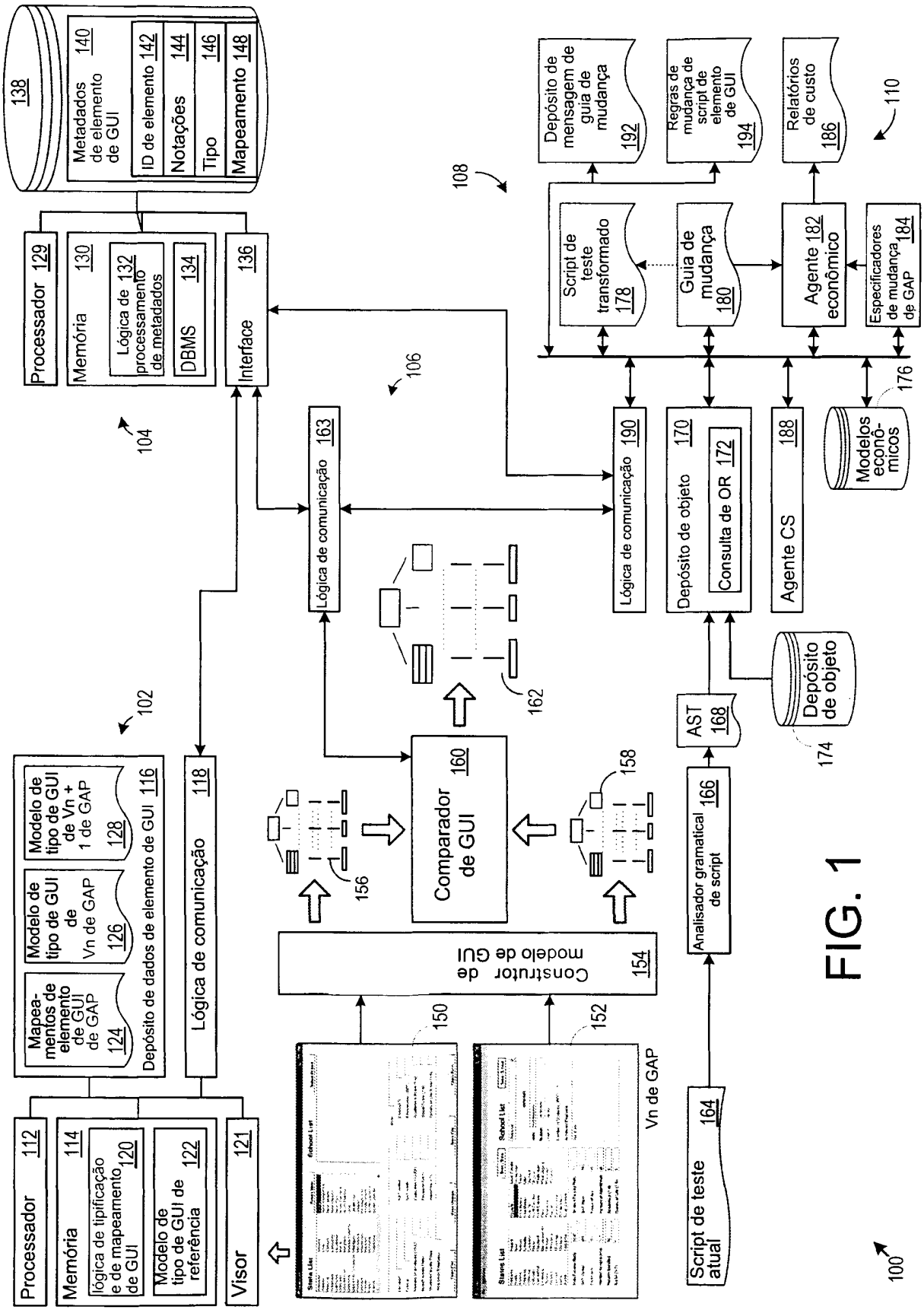

Processador 112
Memória 114
lógica de tipificação e de mapeamento de GUI 120
Modelo de tipo de GUI de referência 122
Visor 121
Mapeamentos de elemento de GUI de GAP 124
Modelo de tipo de GUI de Vn de GAP 126
Modelo de tipo de GUI de Vn + 1 de GAP 128
Depósito de dados de elemento de GUI 116
Lógica de comunicação 118
102
Vn de GAP
150
152
Construtor de modelo de GUI 154
156
158
Comparador de GUI 160
162
Lógica de comunicação 163
Analisador gramatical de script 166
AST 168
Script de teste atual 164
Depósito de objeto 174
Depósito de objeto 170
Consulta de OR 172
Agente CS 188
Lógica de comunicação 190
Modelos econômicos 176
Script de teste transformado 178
Guia de mudança 180
Agente 182 econômico
Especificadores de mudança de GAP 184
Depósito de mensagem de guia de mudança 192
Regras de mudança de script de elemento de GUI 194
Relatórios de custo 186
106
108
110
104
Processador 129
Memória 130
Lógica de processamento de metadados 132
DBMS 134
Interface 136
138
Metadados de elemento de GUI 140
ID de elemento 142
Notações 144
Tipo 146
Mapeamento 148
100

FIG. 1

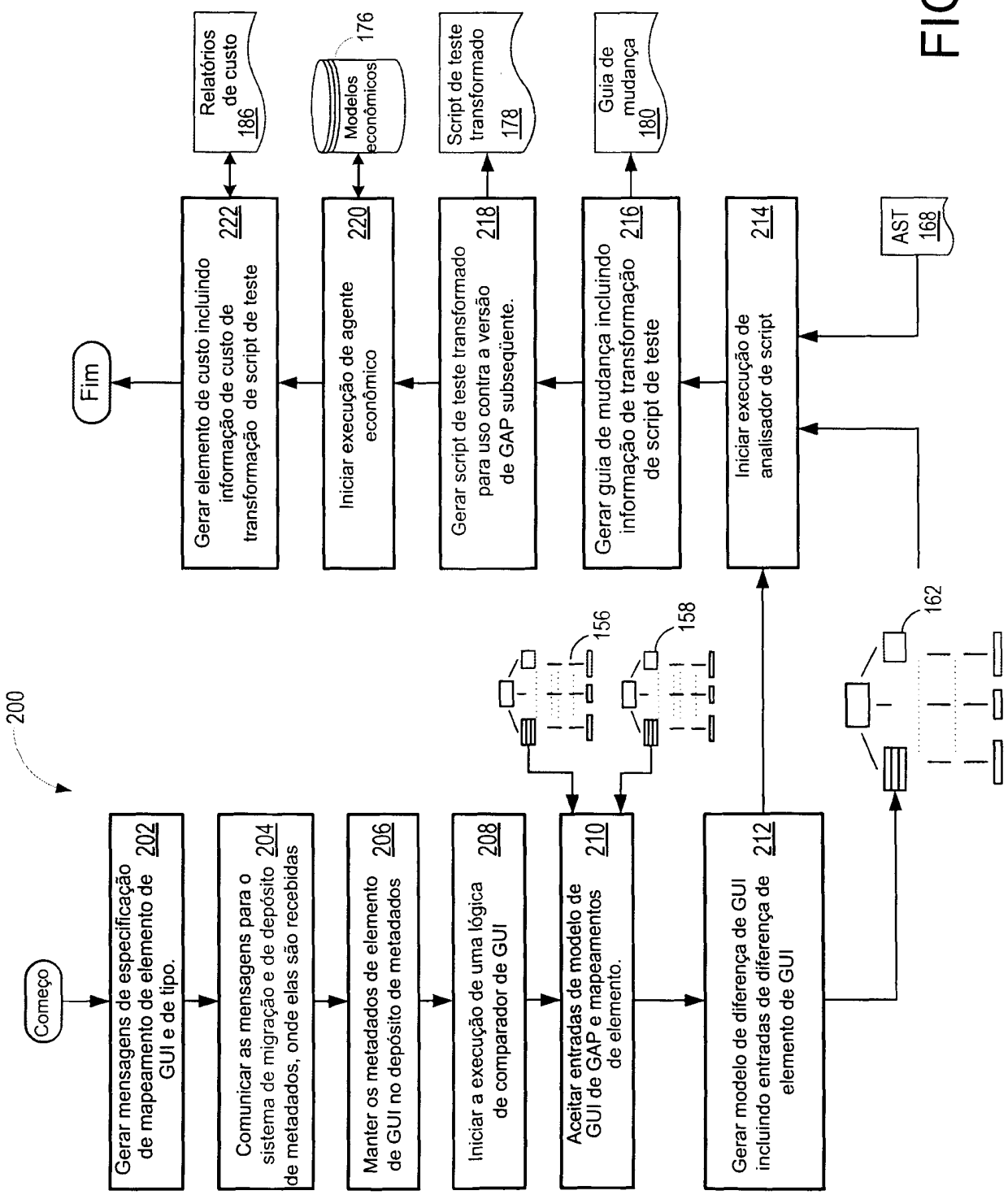

200
Começo
Gerar mensagens de especificação de mapeamento de elemento de GUI e de tipo. 202
Comunicar as mensagens para o sistema de migração e de depósito de metadados, onde elas são recebidas 204
Manter os metadados de elemento de GUI no depósito de metadados 206
Iniciar a execução de uma lógica de comparador de GUI 208
Aceitar entradas de modelo de GUI de GAP e mapeamentos de elemento. 210
Gerar modelo de diferença de GUI incluindo entradas de diferença de elemento de GUI 212
Iniciar execução de analisador de script 214
Gerar guia de mudança incluindo informação de transformação de script de teste 216
Gerar script de teste transformado para uso contra a versão de GAP subseqüente. 218
Iniciar execução de agente econômico 220
Gerar elemento de custo incluindo informação de custo de transformação de script de teste 222
Fim
AST 168
Guia de mudança 180
Script de teste transformado 178
Modelos econômicos 176
Relatórios de custo 186
156
158
162

FIG. 2

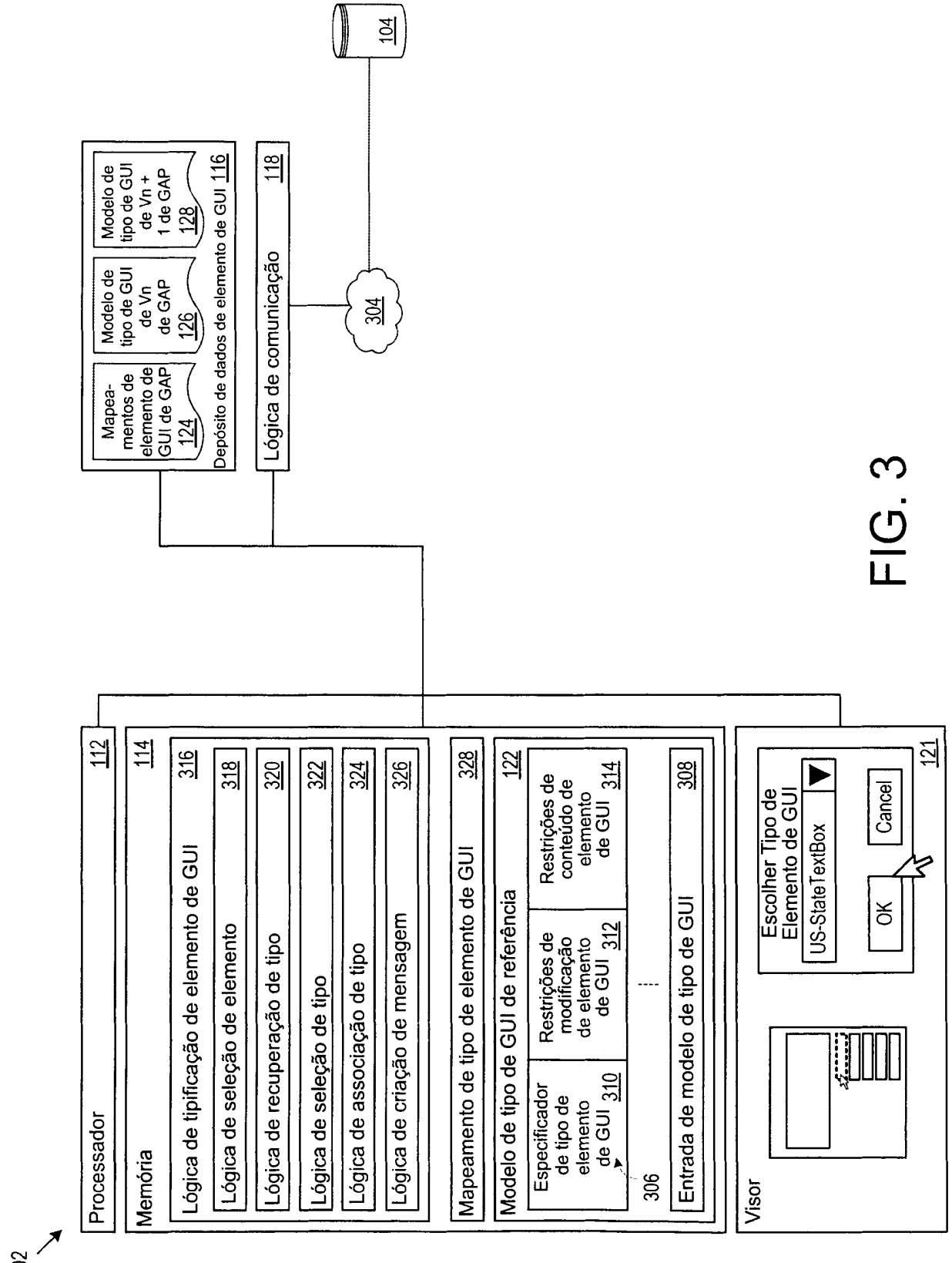

102
Processador 112
Memória 114
Lógica de tipificação de elemento de GUI 316
Lógica de seleção de elemento 318
Lógica de recuperação de tipo 320
Lógica de seleção de tipo 322
Lógica de associação de tipo 324
Lógica de criação de mensagem 326
Mapeamento de tipo de elemento de GUI 328
Modelo de tipo de GUI de referência 122
Especificador de tipo de elemento de GUI 310
Restrições de modificação de elemento de GUI 312
Restrições de conteúdo de elemento de GUI 314
306
Entrada de modelo de tipo de GUI 308
Visor 121
Escolher Tipo de Elemento de GUI
US-StateTextBox
OK
Cancel
Depósito de dados de elemento de GUI 116
Mapeamentos de elemento de GUI de GAP 124
Modelo de tipo de GUI de Vn de GAP 126
Modelo de tipo de GUI de Vn + 1 de GAP 128
Lógica de comunicação 118
304
104

FIG. 3

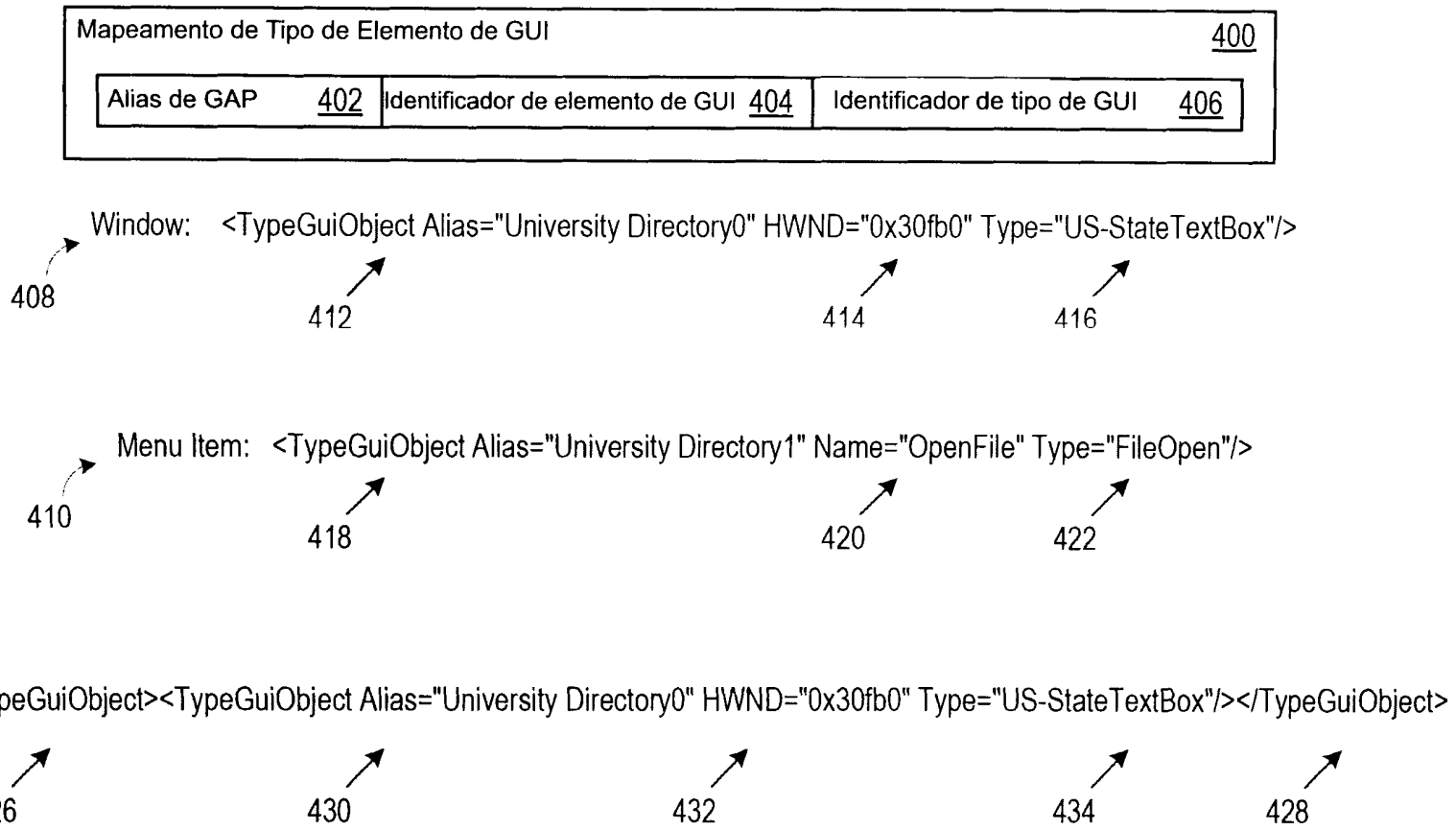

Mapeamento de Tipo de Elemento de GUI
400
Alias de GAP
402
Identificador de elemento de GUI
404
Identificador de tipo de GUI
406
Window: <TypeGuiObject Alias="University Directory0" HWND="0x30fb0" Type="US-StateTextBox"/>
408
412
414
416
Menu Item: <TypeGuiObject Alias="University Directory1" Name="OpenFile" Type="FileOpen"/>
410
418
420
422
<TypeGuiObject><TypeGuiObject Alias="University Directory0" HWND="0x30fb0" Type="US-StateTextBox"/></TypeGuiObject>
424
426
430
432
434
428

FIG. 4

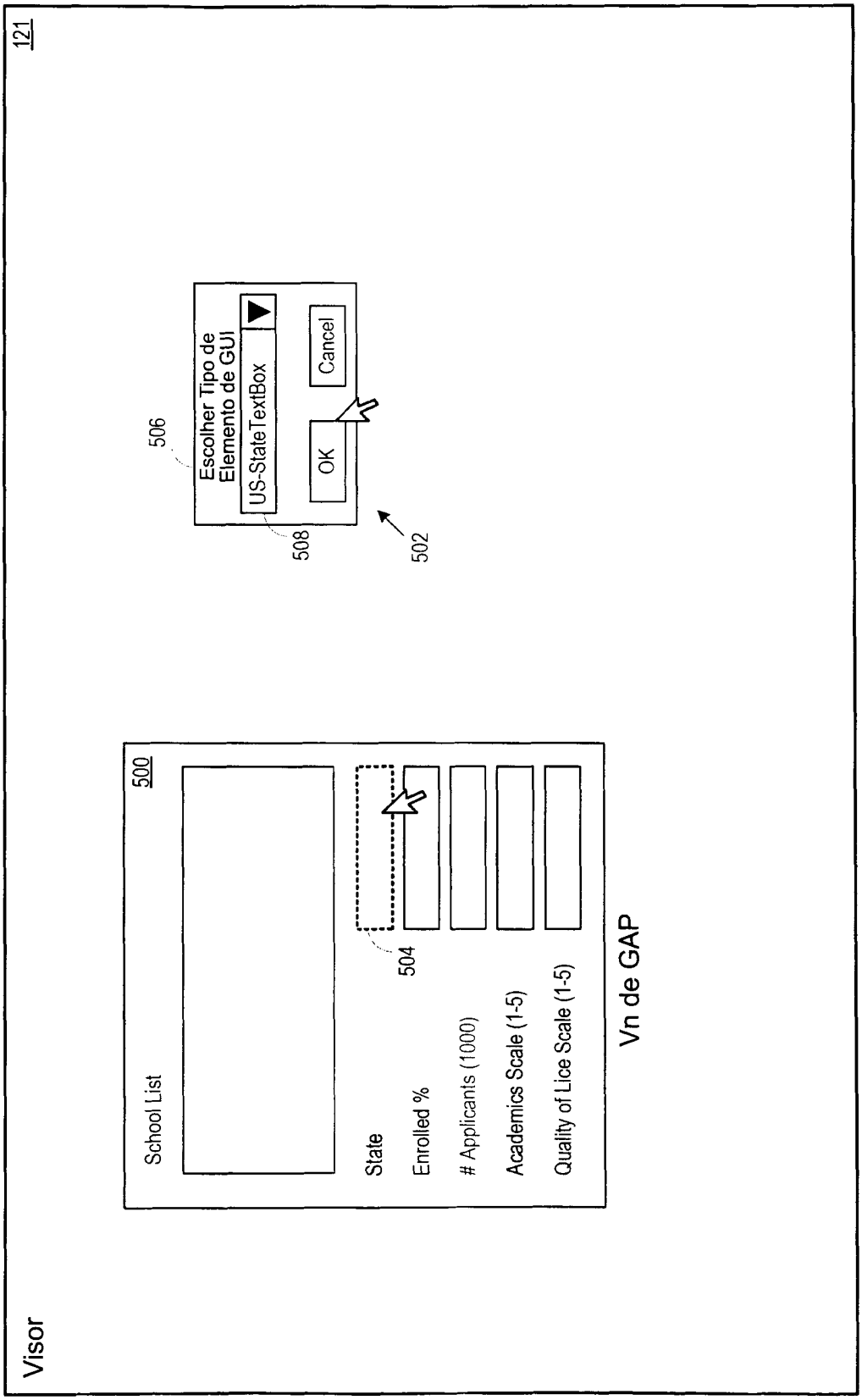

121
Visor
Escolher Tipo de Elemento de GUI
US-StateTextBox
OK
Cancel
506
508
502
500
School List
State
Enrolled %
Applicants (1000)
Academics Scale (1-5)
Quality of Lice Scale (1-5)
504
Vn de GAP

FIG. 5

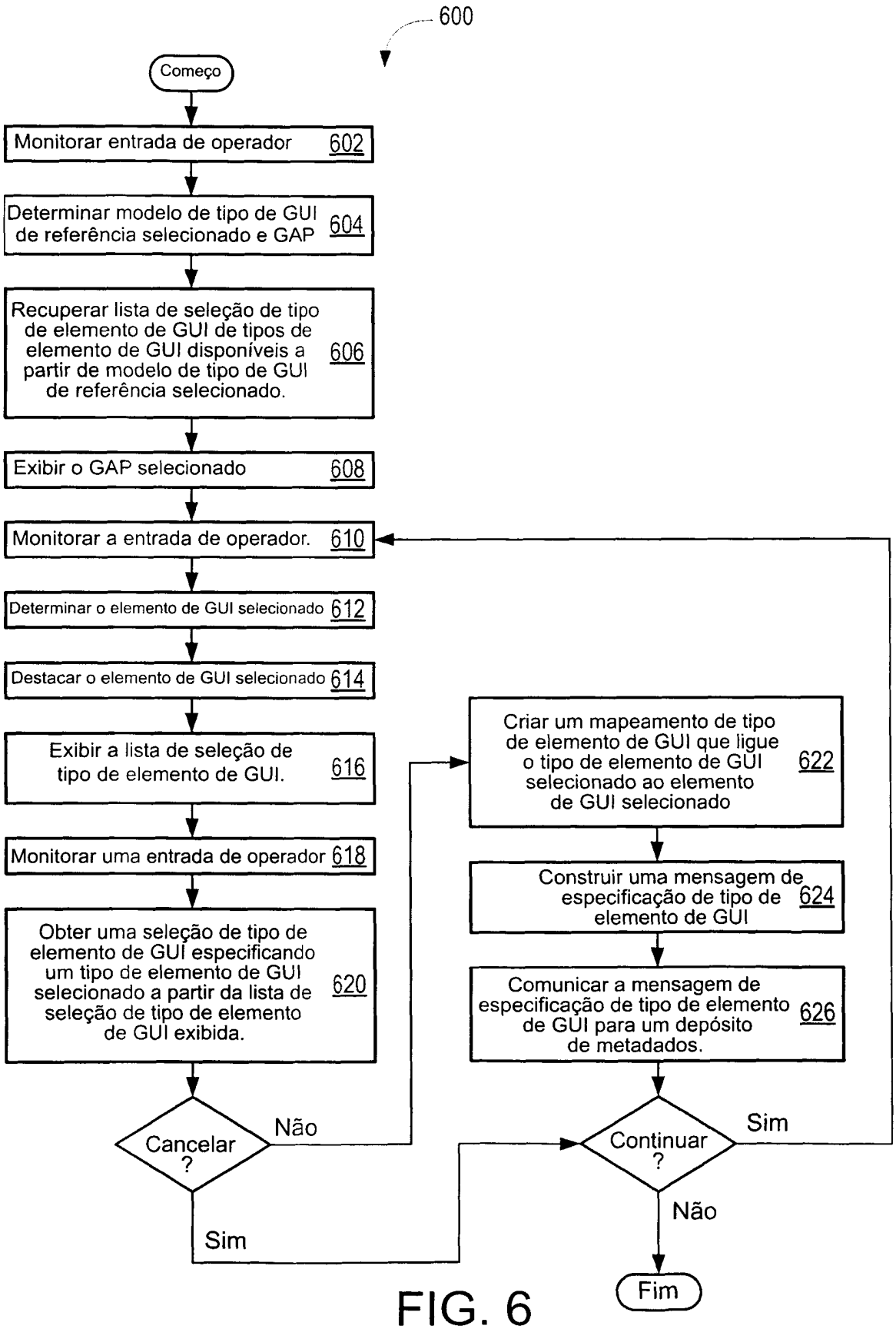

600
Começo
Monitorar entrada de operador 602
Determinar modelo de tipo de GUI de referência selecionado e GAP 604
Recuperar lista de seleção de tipo de elemento de GUI de tipos de elemento de GUI disponíveis a partir de modelo de tipo de GUI de referência selecionado. 606
Exibir o GAP selecionado 608
Monitorar a entrada de operador. 610
Determinar o elemento de GUI selecionado 612
Destacar o elemento de GUI selecionado 614
Exibir a lista de seleção de tipo de elemento de GUI. 616
Monitorar uma entrada de operador 618
Obter uma seleção de tipo de elemento de GUI especificando um tipo de elemento de GUI selecionado a partir da lista de seleção de tipo de elemento de GUI exibida. 620
Cancelar ?
Não
Sim
Criar um mapeamento de tipo de elemento de GUI que ligue o tipo de elemento de GUI selecionado ao elemento de GUI selecionado 622
Construir uma mensagem de especificação de tipo de elemento de GUI 624
Comunicar a mensagem de especificação de tipo de elemento de GUI para um depósito de metadados. 626
Continuar ?
Sim
Não
Fim

FIG. 6

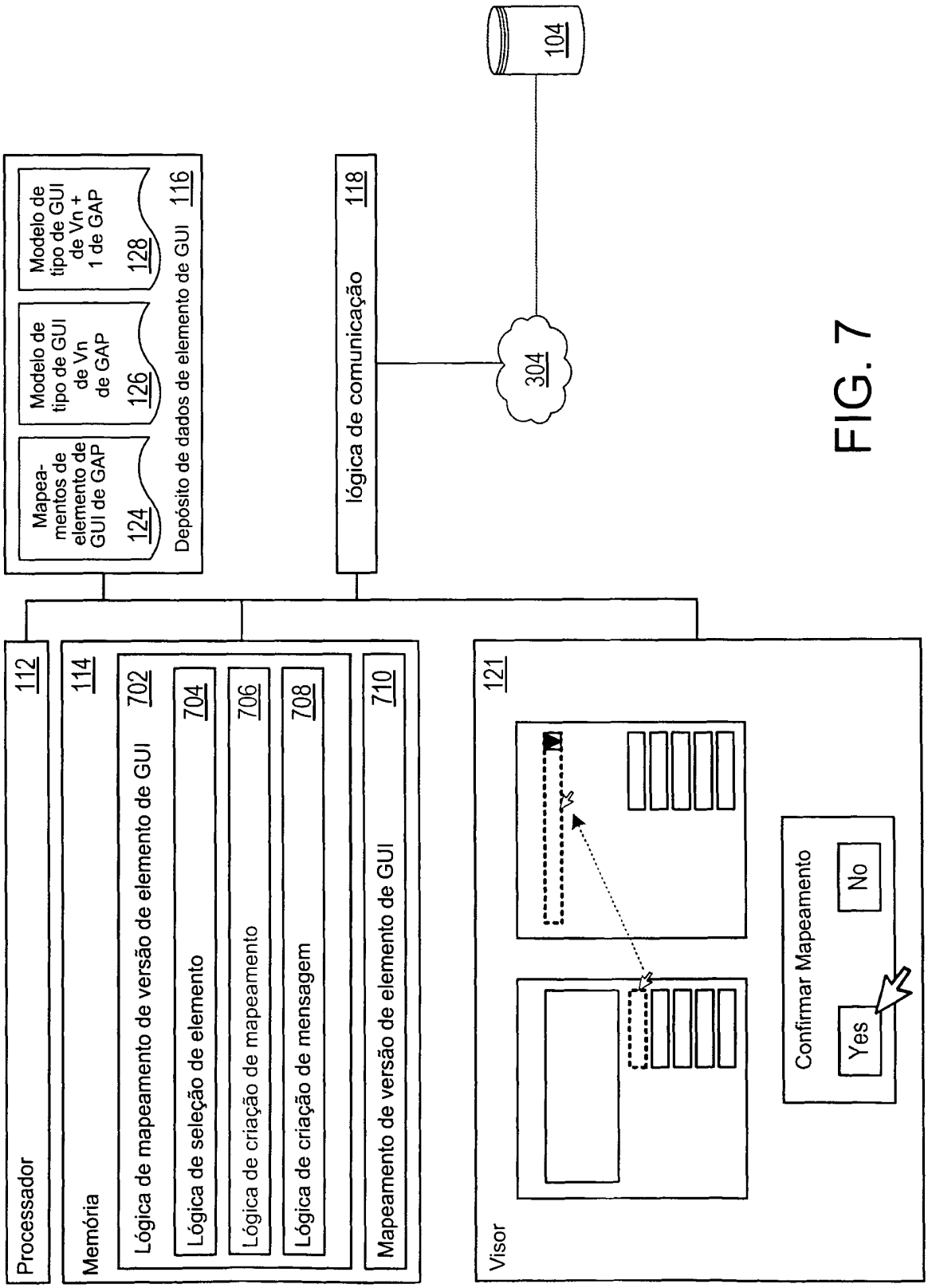

Processador 112
Memória 114
Lógica de mapeamento de versão de elemento de GUI 702
Lógica de seleção de elemento 704
Lógica de criação de mapeamento 706
Lógica de criação de mensagem 708
Mapeamento de versão de elemento de GUI 710
Depósito de dados de elemento de GUI 116
Mapea-mentos de elemento de GUI de GAP 124
Modelo de tipo de GUI de Vn de GAP 126
Modelo de tipo de GUI de Vn + 1 de GAP 128
lógica de comunicação 118
304
104
Visor 121
Confirmar Mapeamento
Yes
No

FIG. 7

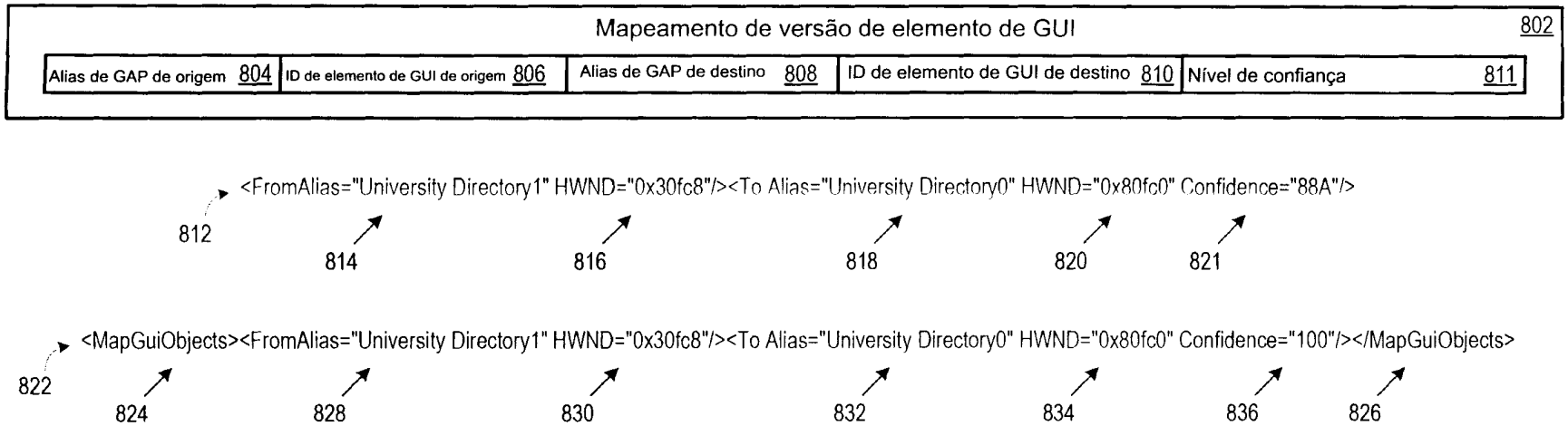

Mapeamento de versão de elemento de GUI
802
Alias de GAP de origem 804
ID de elemento de GUI de origem 806
Alias de GAP de destino 808
ID de elemento de GUI de destino 810
Nível de confiança 811
812
<FromAlias="University Directory1" HWND="0x30fc8"/><To Alias="University Directory0" HWND="0x80fc0" Confidence="88A"/>
814
816
818
820
821
822
<MapGuiObjects><FromAlias="University Directory1" HWND="0x30fc8"/><To Alias="University Directory0" HWND="0x80fc0" Confidence="100"/></MapGuiObjects>
824
828
830
832
834
836
826

FIG. 8

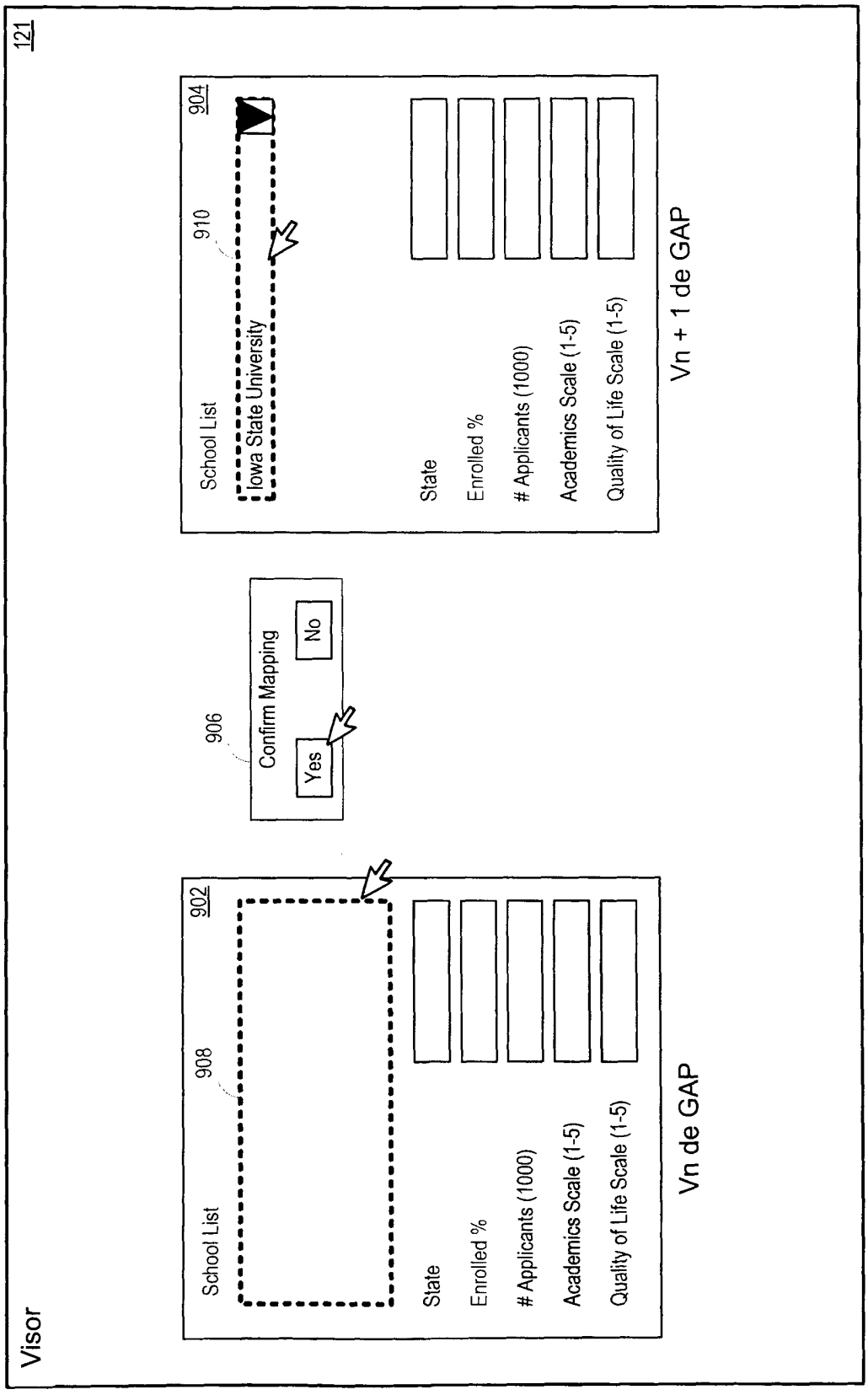

121
Visor
School List
908
902
State
Enrolled %
Applicants (1000)
Academics Scale (1-5)
Quality of Life Scale (1-5)
Vn de GAP
906
Confirm Mapping
Yes
No
School List
910
904
Iowa State University
State
Enrolled %
Applicants (1000)
Academics Scale (1-5)
Quality of Life Scale (1-5)
Vn + 1 de GAP

FIG. 9

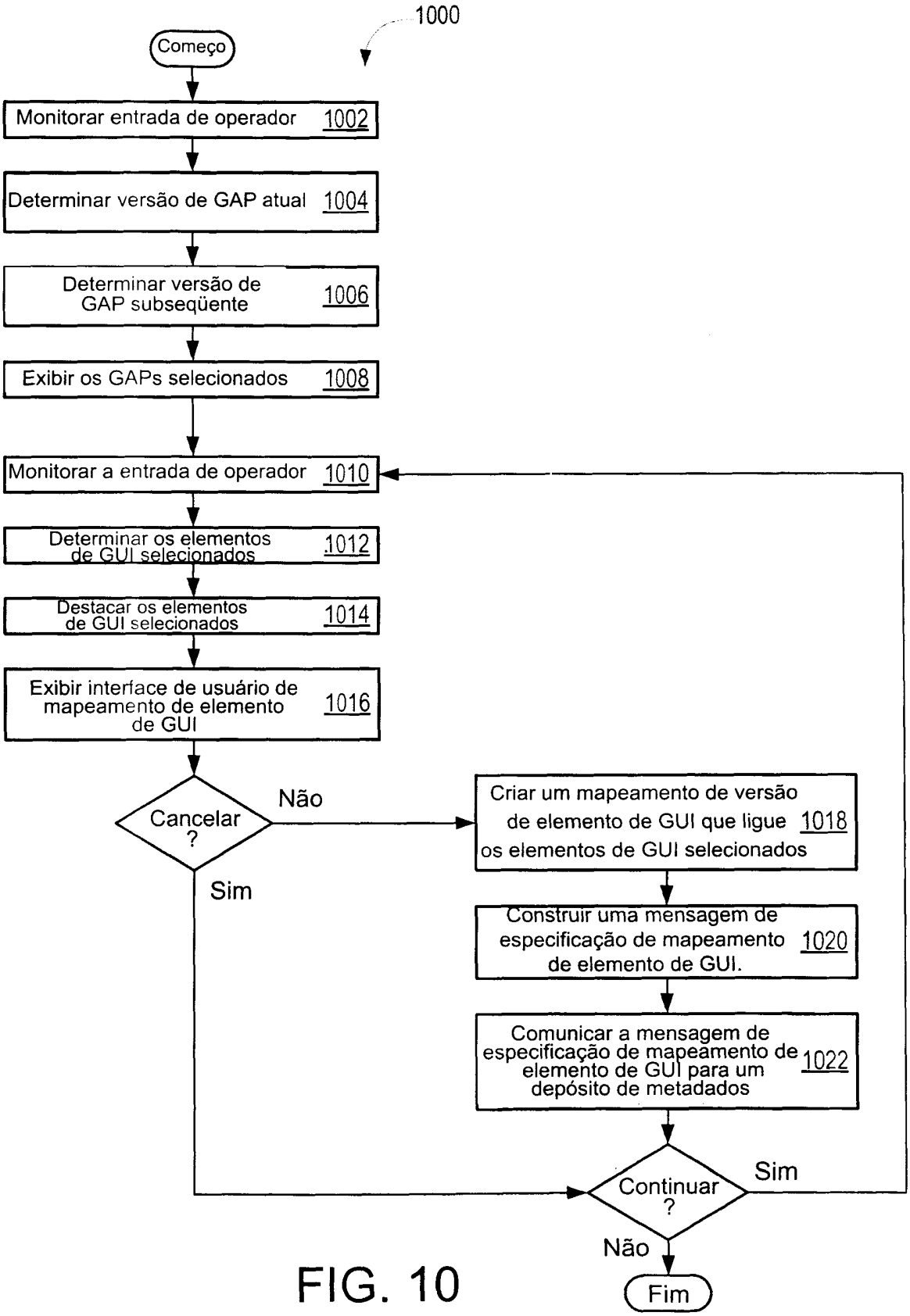

1000
Começo
Monitorar entrada de operador 1002
Determinar versão de GAP atual 1004
Determinar versão de GAP subseqüente 1006
Exibir os GAPs selecionados 1008
Monitorar a entrada de operador 1010
Determinar os elementos de GUI selecionados 1012
Destacar os elementos de GUI selecionados 1014
Exibir interface de usuário de mapeamento de elemento de GUI 1016
Cancelar ?
Não
Sim
Criar um mapeamento de versão de elemento de GUI que ligue os elementos de GUI selecionados 1018
Construir uma mensagem de especificação de mapeamento de elemento de GUI. 1020
Comunicar a mensagem de especificação de mapeamento de elemento de GUI para um depósito de metadados 1022
Continuar ?
Sim
Não
Fim

FIG. 10

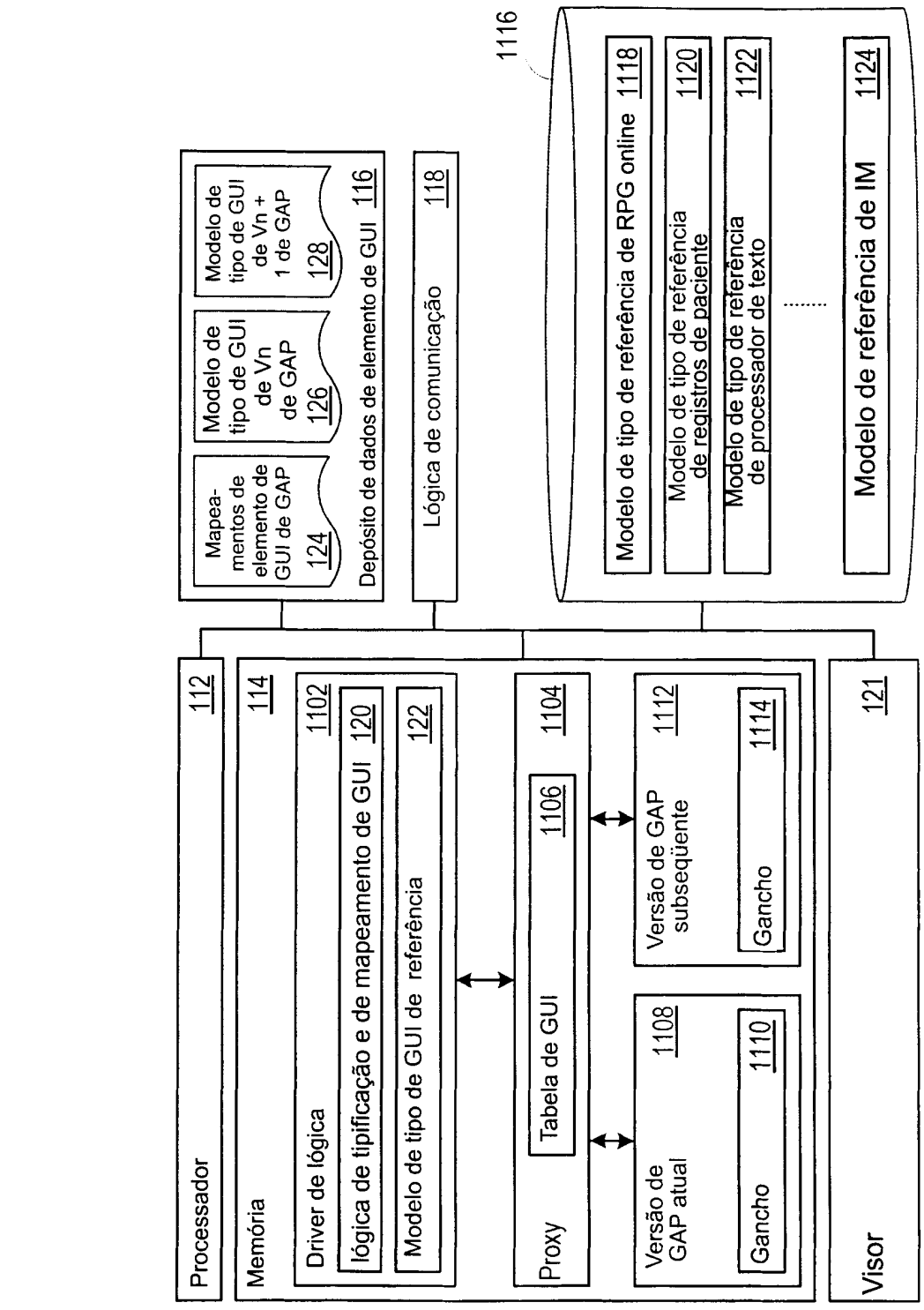

1100
Processador 112
Memória 114
Driver de lógica 1102
lógica de tipificação e de mapeamento de GUI 120
Modelo de tipo de GUI de referência 122
Proxy
Tabela de GUI 1106
1104
Versão de GAP atual 1108
Gancho 1110
Versão de GAP subseqüente 1112
Gancho 1114
Visor 121
Depósito de dados de elemento de GUI 116
Mapea-mentos de elemento de GUI de GAP 124
Modelo de tipo de GUI de Vn de GAP 126
Modelo de tipo de GUI de Vn + 1 de GAP 128
Lógica de comunicação 118
1116
Modelo de tipo de referência de RPG online 1118
Modelo de tipo de referência de registros de paciente 1120
Modelo de tipo de referência de processador de texto 1122
Modelo de referência de IM 1124

FIG. 11

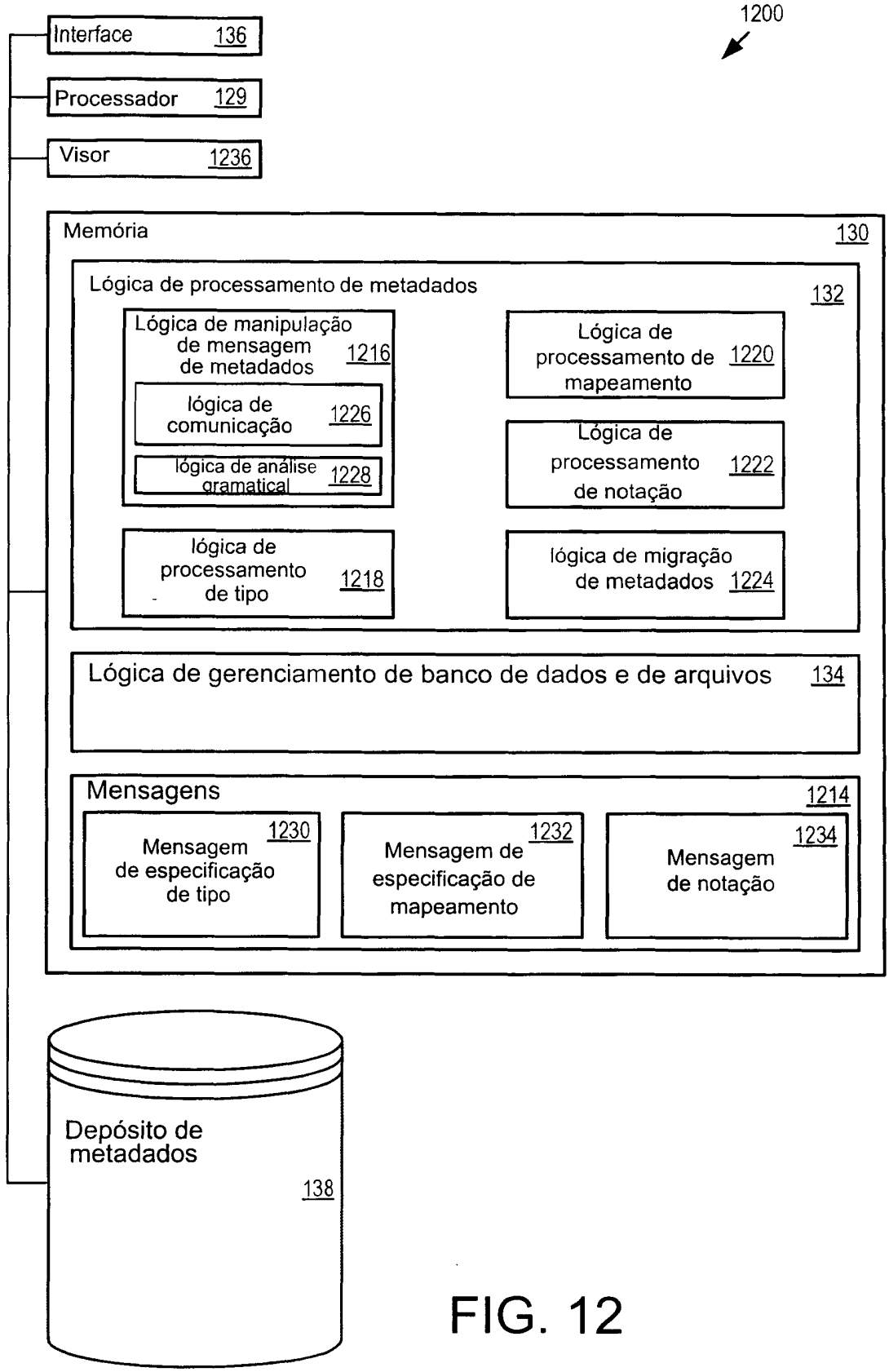

1200
Interface 136
Processador 129
Visor 1236
Memória 130
Lógica de processamento de metadados 132
Lógica de manipulação de mensagem de metadados 1216
lógica de comunicação 1226
lógica de análise gramatical 1228
Lógica de processamento de mapeamento 1220
Lógica de processamento de notação 1222
lógica de processamento de tipo 1218
lógica de migração de metadados 1224
Lógica de gerenciamento de banco de dados e de arquivos 134
Mensagens 1214
Mensagem de especificação de tipo 1230
Mensagem de especificação de mapeamento 1232
Mensagem de notação 1234
Depósito de metadados 138

FIG. 12

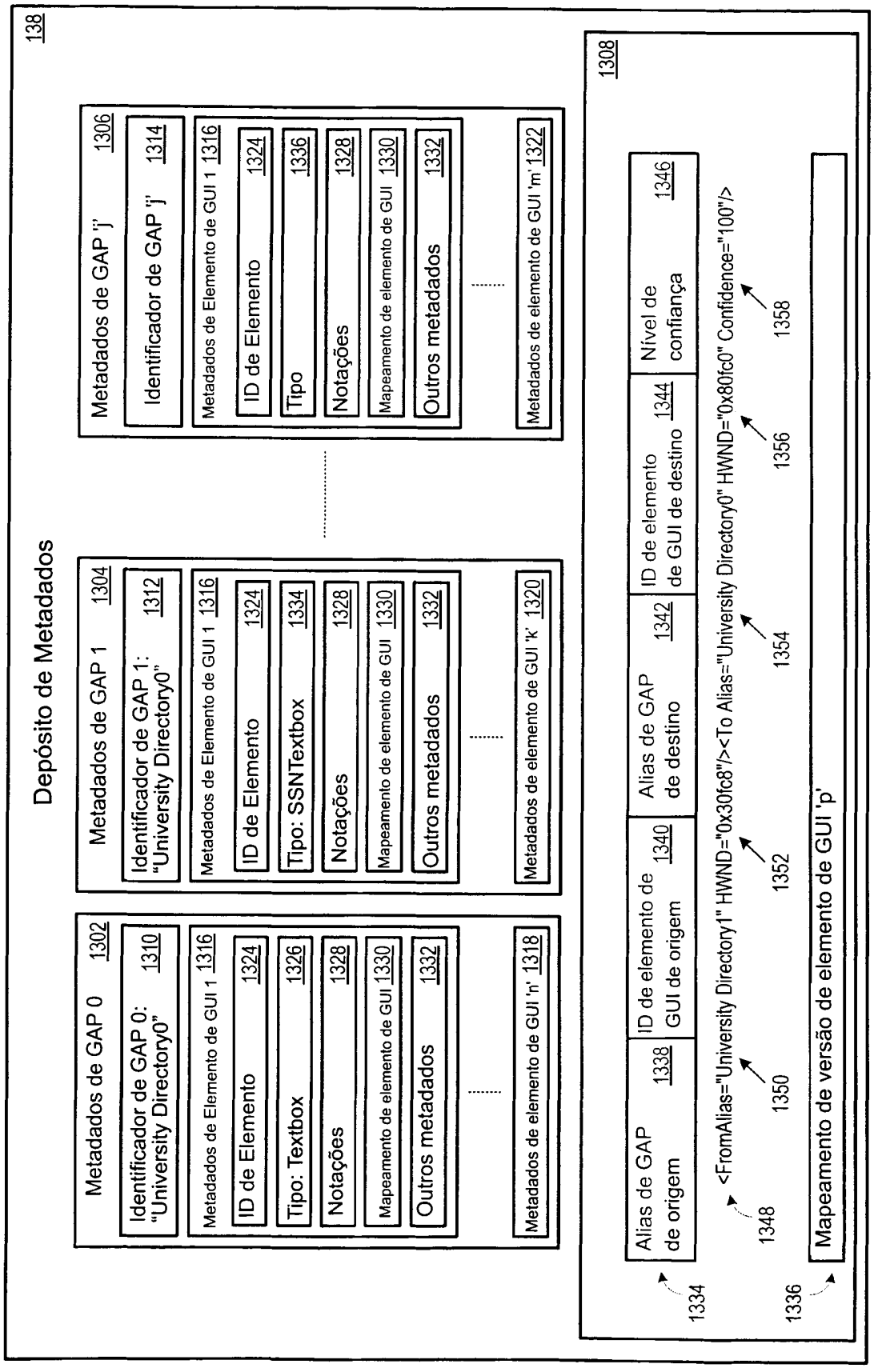
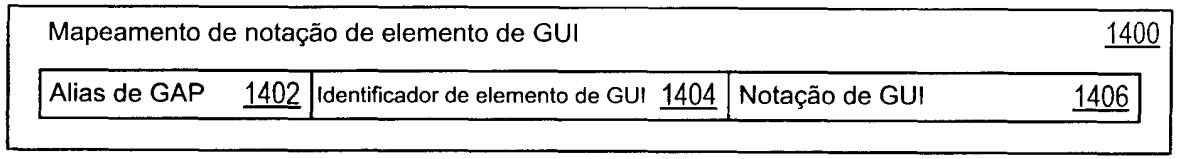

138
Depósito de Metadados
Metadados de GAP 0 1302
Identificador de GAP 0: "University Directory0" 1310
Metadados de Elemento de GUI 1 1316
ID de Elemento 1324
Tipo: Textbox 1326
Notações 1328
Mapeamento de elemento de GUI 1330
Outros metadados 1332
Metadados de elemento de GUI 'n' 1318
Metadados de GAP 1 1304
Identificador de GAP 1: "University Directory0" 1312
Metadados de Elemento de GUI 1 1316
ID de Elemento 1324
Tipo: SSNTextbox 1334
Notações 1328
Mapeamento de elemento de GUI 1330
Outros metadados 1332
Metadados de elemento de GUI 'k' 1320
Metadados de GAP 'j' 1306
Identificador de GAP 'j' 1314
Metadados de Elemento de GUI 1 1316
ID de Elemento 1324
Tipo 1336
Notações 1328
Mapeamento de elemento de GUI 1330
Outros metadados 1332
Metadados de elemento de GUI 'm' 1322
1308
Alias de GAP de origem 1338
ID de elemento de GUI de origem 1340
Alias de GAP de destino 1342
ID de elemento de GUI de destino 1344
Nível de confiança 1346
1334
<FromAlias="University Directory1" HWND="0x30fc8"/><To Alias="University Directory0" HWND="0x80fc0" Confidence="100"/>
1348
1350
1352
1354
1356
1358
1336
Mapeamento de versão de elemento de GUI 'p'

FIG. 13



1408 Window: <NotationGuiObject Alias="University Directory0" HWND="0x30fb0" Notation="State Names Only"/>

1412 1414 1416

1410 Menu Item: <NotationGuiObject Alias="University Directory1" Name="OpenFile" Notation="Opens to Default Directory"/>

1418 1420 1422

1424 <NotationGuiObject><NotationGuiObject Alias="University Directory0" HWND="0x30fb0" Notation="State Names Only"/></NotationGuiObject>

1426 1430 1432 1434 1428

FIG. 14

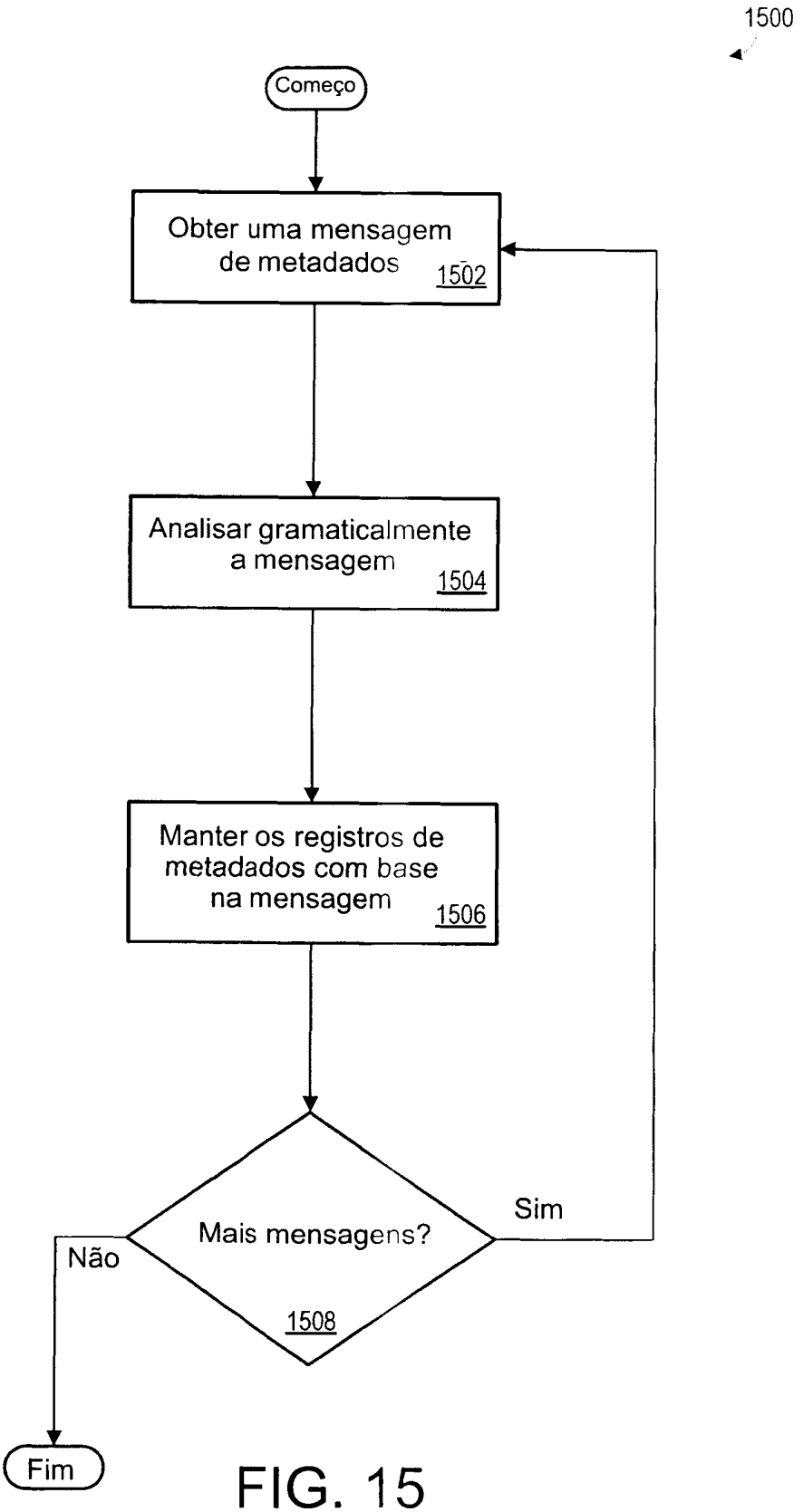
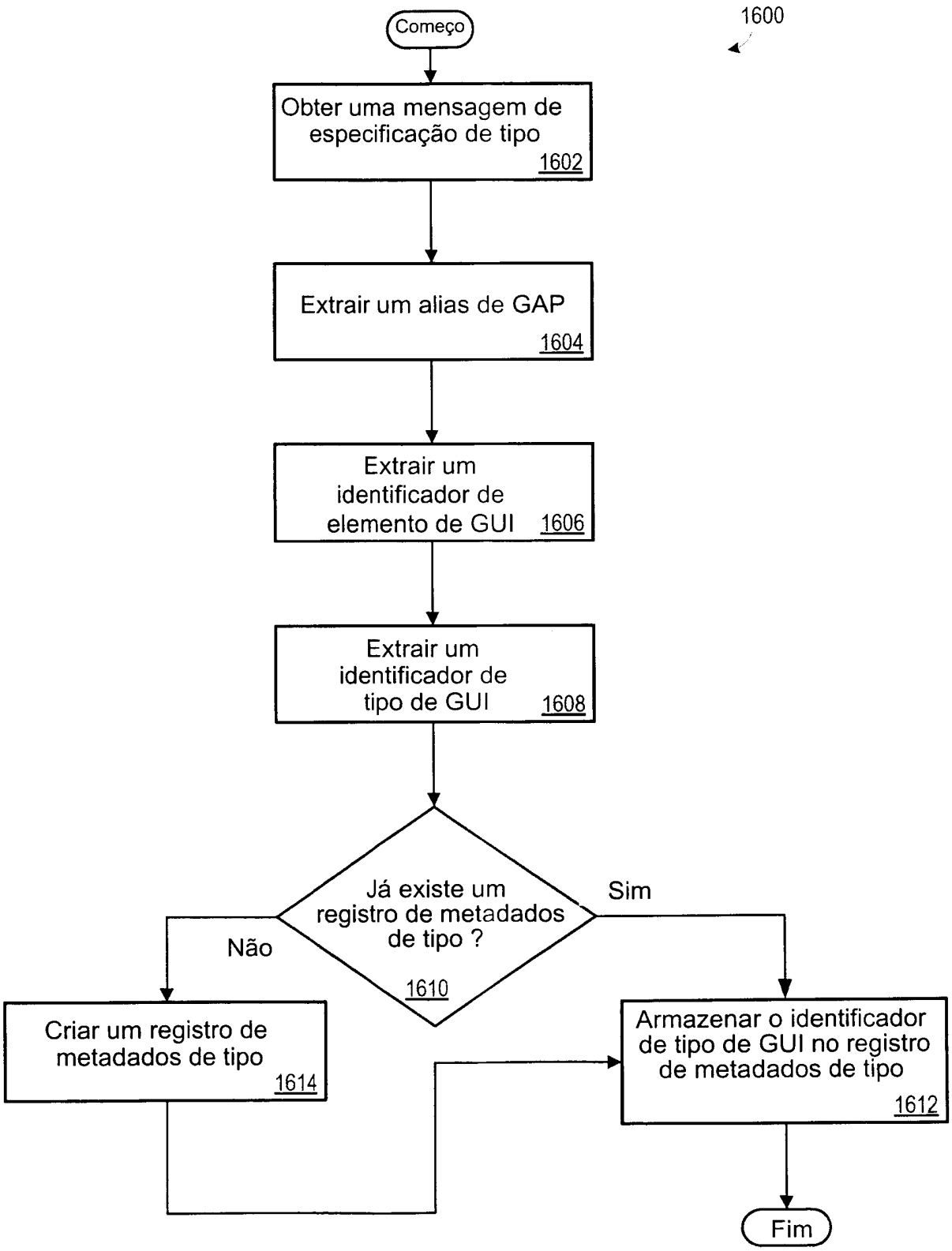

1500
Começo
Obter uma mensagem de metadados
1502
Analisar gramaticalmente a mensagem
1504
Manter os registros de metadados com base na mensagem
1506
Mais mensagens?
1508
Sim
Não
Fim

FIG. 15


1600
Começo
Obter uma mensagem de especificação de tipo
1602
Extrair um alias de GAP
1604
Extrair um identificador de elemento de GUI
1606
Extrair um identificador de tipo de GUI
1608
Já existe um registro de metadados de tipo ?
1610
Sim
Não
Criar um registro de metadados de tipo
1614
Armazenar o identificador de tipo de GUI no registro de metadados de tipo
1612
Fim

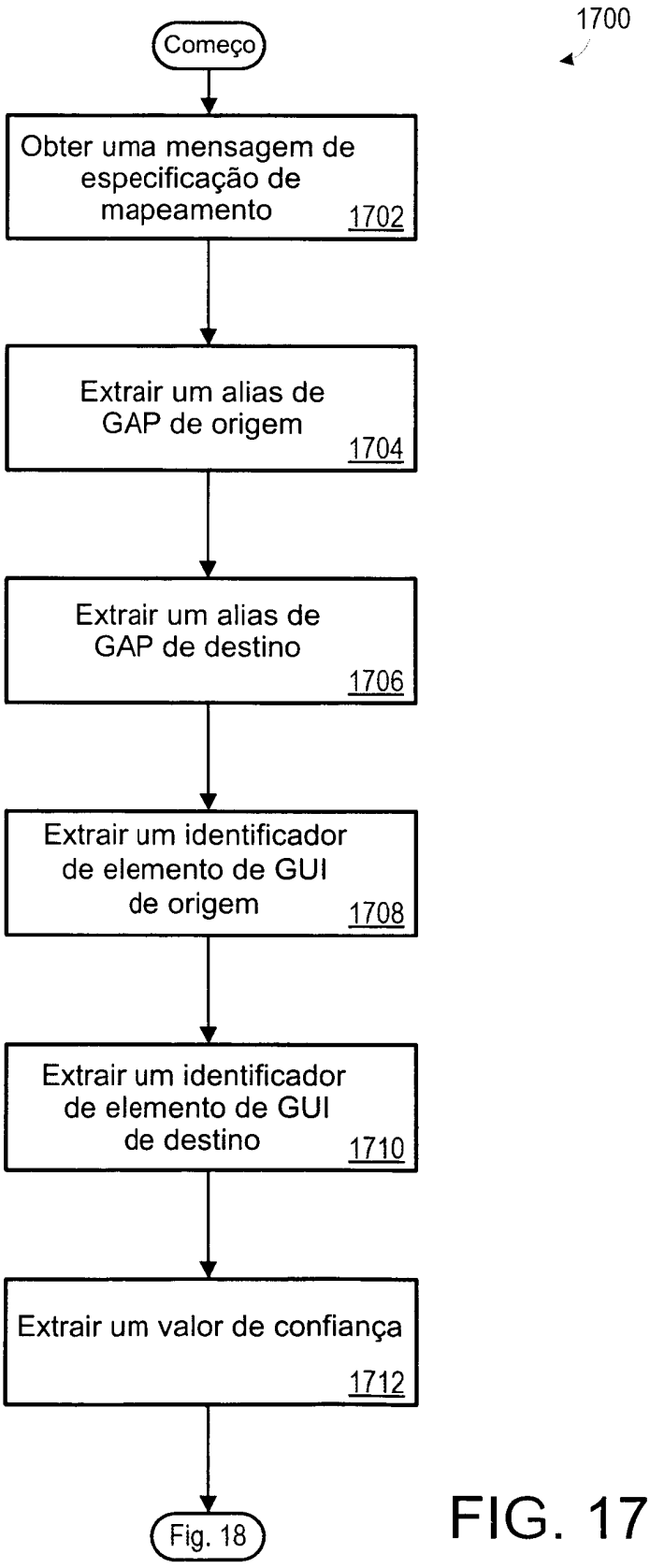

1700
Começo
Obter uma mensagem de especificação de mapeamento
1702
Extrair um alias de GAP de origem
1704
Extrair um alias de GAP de destino
1706
Extrair um identificador de elemento de GUI de origem
1708
Extrair um identificador de elemento de GUI de destino
1710
Extrair um valor de confiança
1712
Fig. 18

FIG. 17

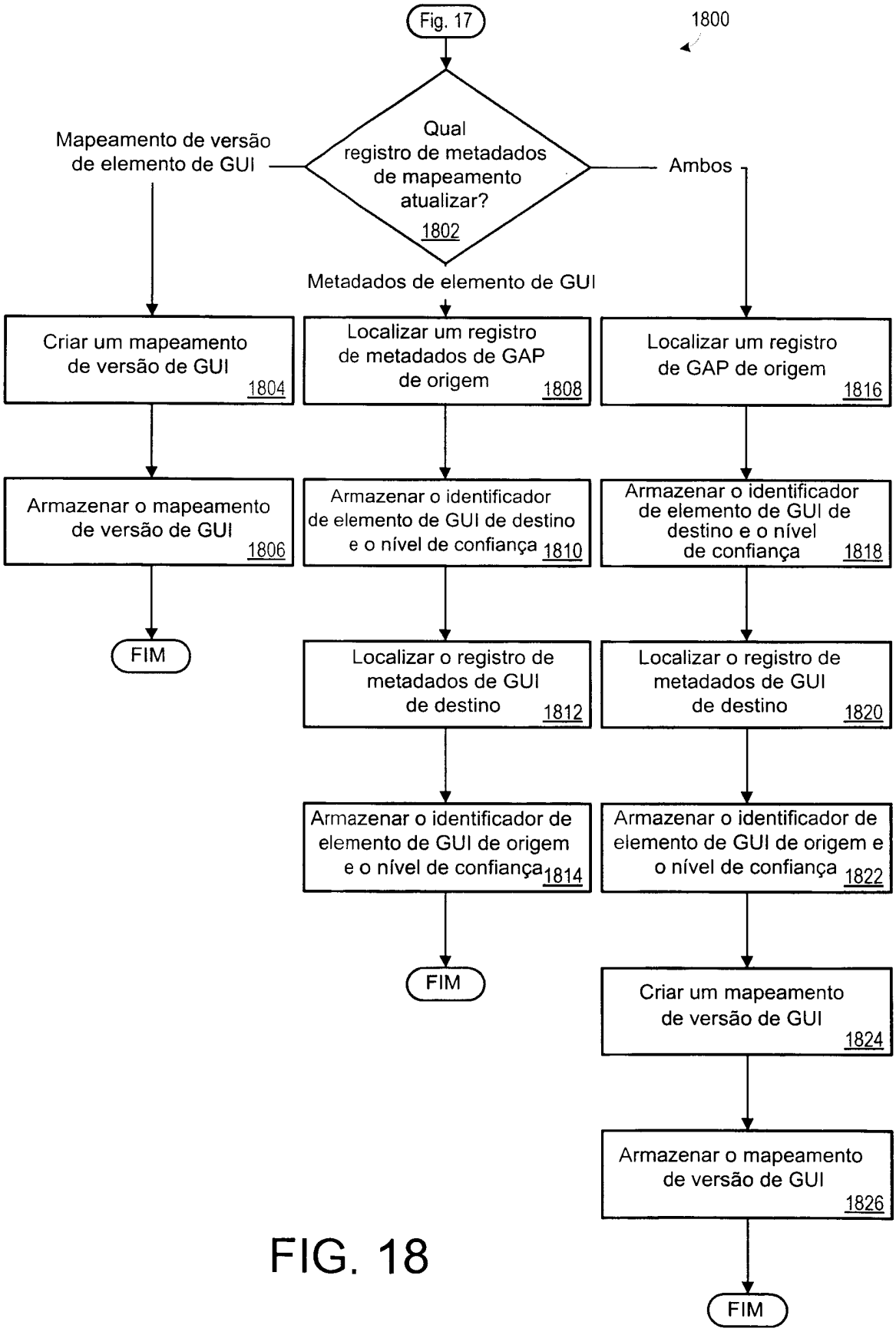


Fig. 17
1800
Qual registro de metadados de mapeamento atualizar? 1802
Mapeamento de versão de elemento de GUI
Metadados de elemento de GUI
Ambos
Criar um mapeamento de versão de GUI 1804
Armazenar o mapeamento de versão de GUI 1806
FIM
Localizar um registro de metadados de GAP de origem 1808
Armazenar o identificador de elemento de GUI de destino e o nível de confiança 1810
Localizar o registro de metadados de GUI de destino 1812
Armazenar o identificador de elemento de GUI de origem e o nível de confiança 1814
FIM
Localizar um registro de GAP de origem 1816
Armazenar o identificador de elemento de GUI de destino e o nível de confiança 1818
Localizar o registro de metadados de GUI de destino 1820
Armazenar o identificador de elemento de GUI de origem e o nível de confiança 1822
Criar um mapeamento de versão de GUI 1824
Armazenar o mapeamento de versão de GUI 1826
FIM

FIG. 18

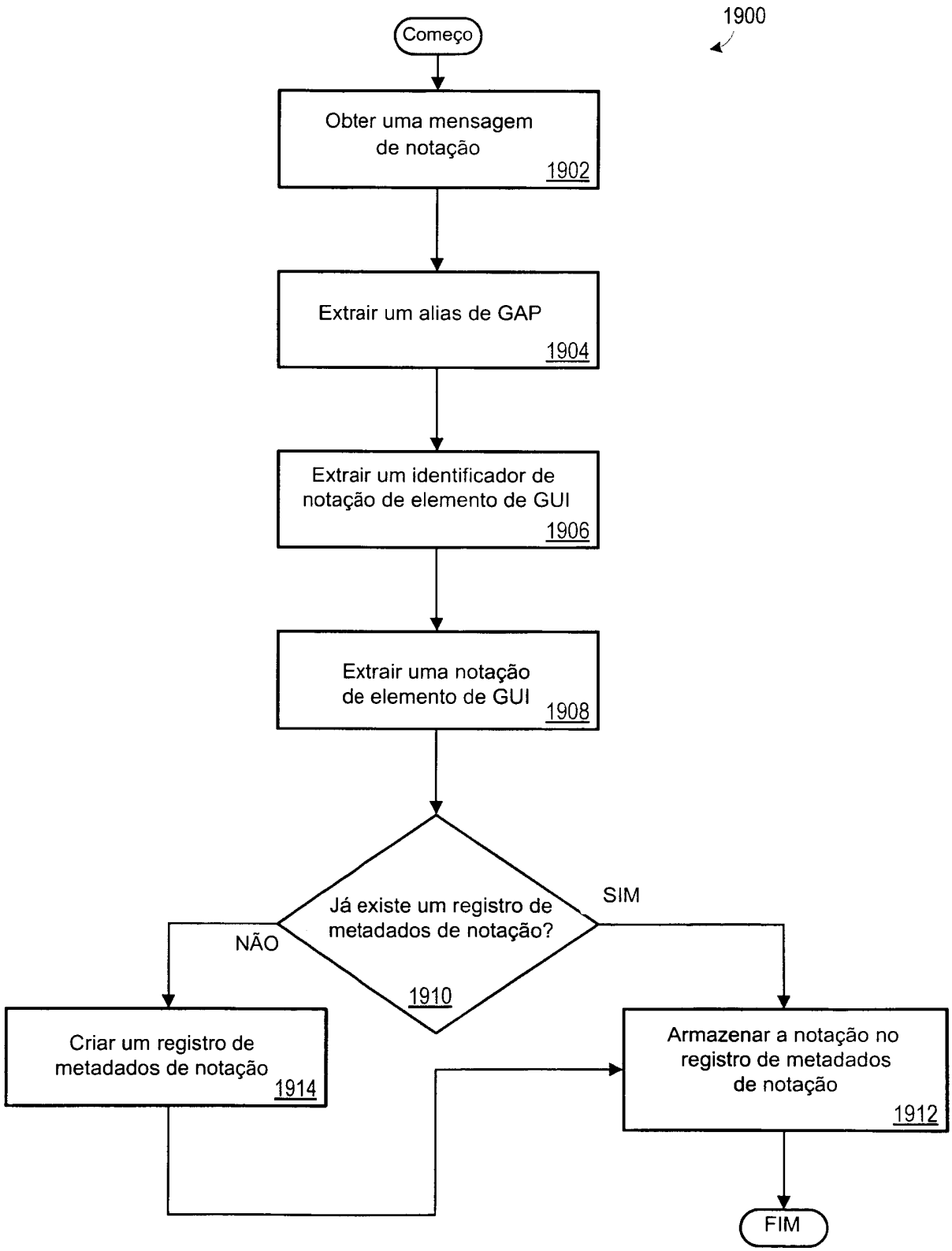

1900
Começo
Obter uma mensagem de notação
1902
Extrair um alias de GAP
1904
Extrair um identificador de notação de elemento de GUI
1906
Extrair uma notação de elemento de GUI
1908
Já existe um registro de metadados de notação?
1910
SIM
NÃO
Criar um registro de metadados de notação
1914
Armazenar a notação no registro de metadados de notação
1912
FIM

FIG. 19

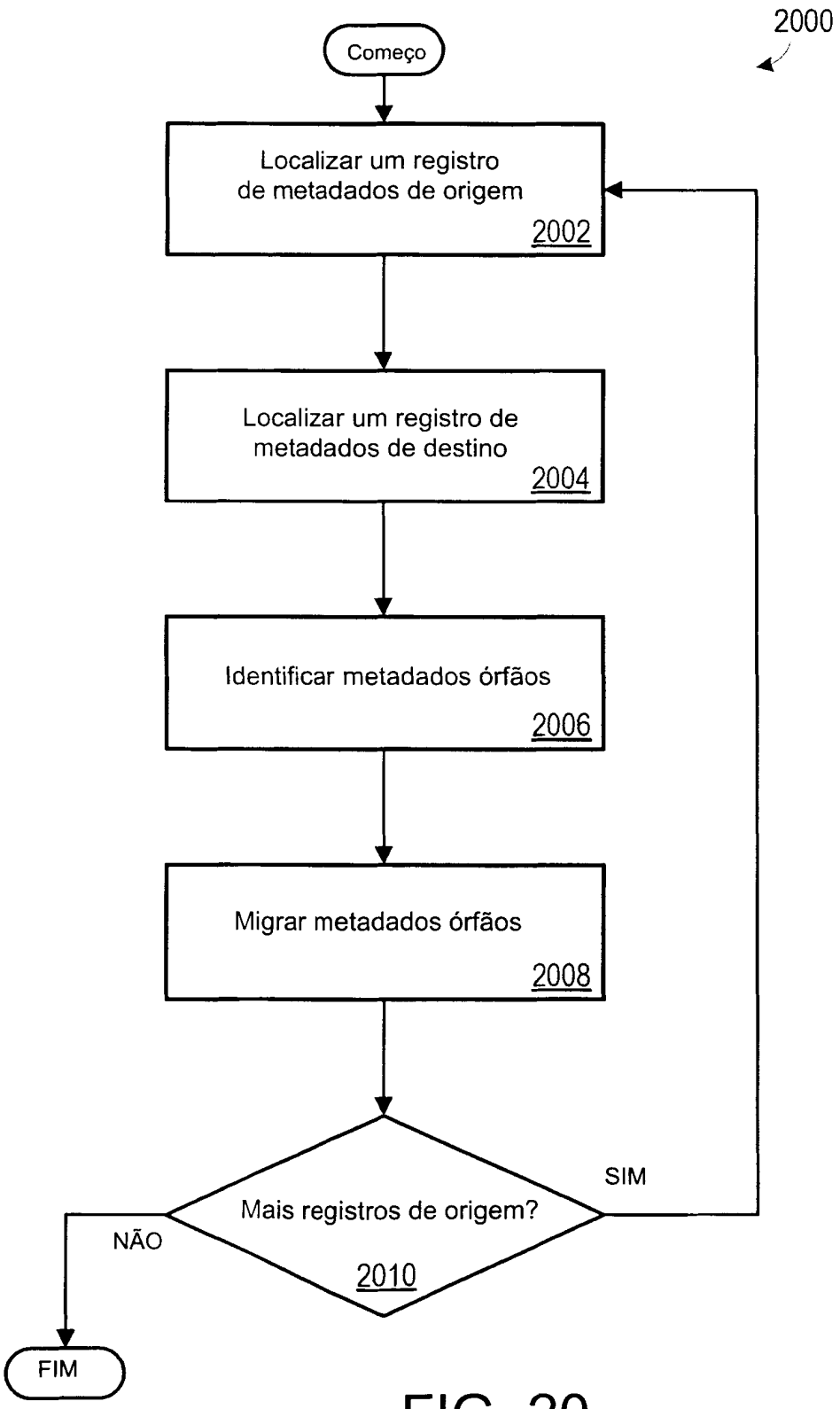

2000
Começo
Localizar um registro de metadados de origem
2002
Localizar um registro de metadados de destino
2004
Identificar metadados órfãos
2006
Migrar metadados órfãos
2008
Mais registros de origem?
2010
SIM
NÃO
FIM

FIG. 20

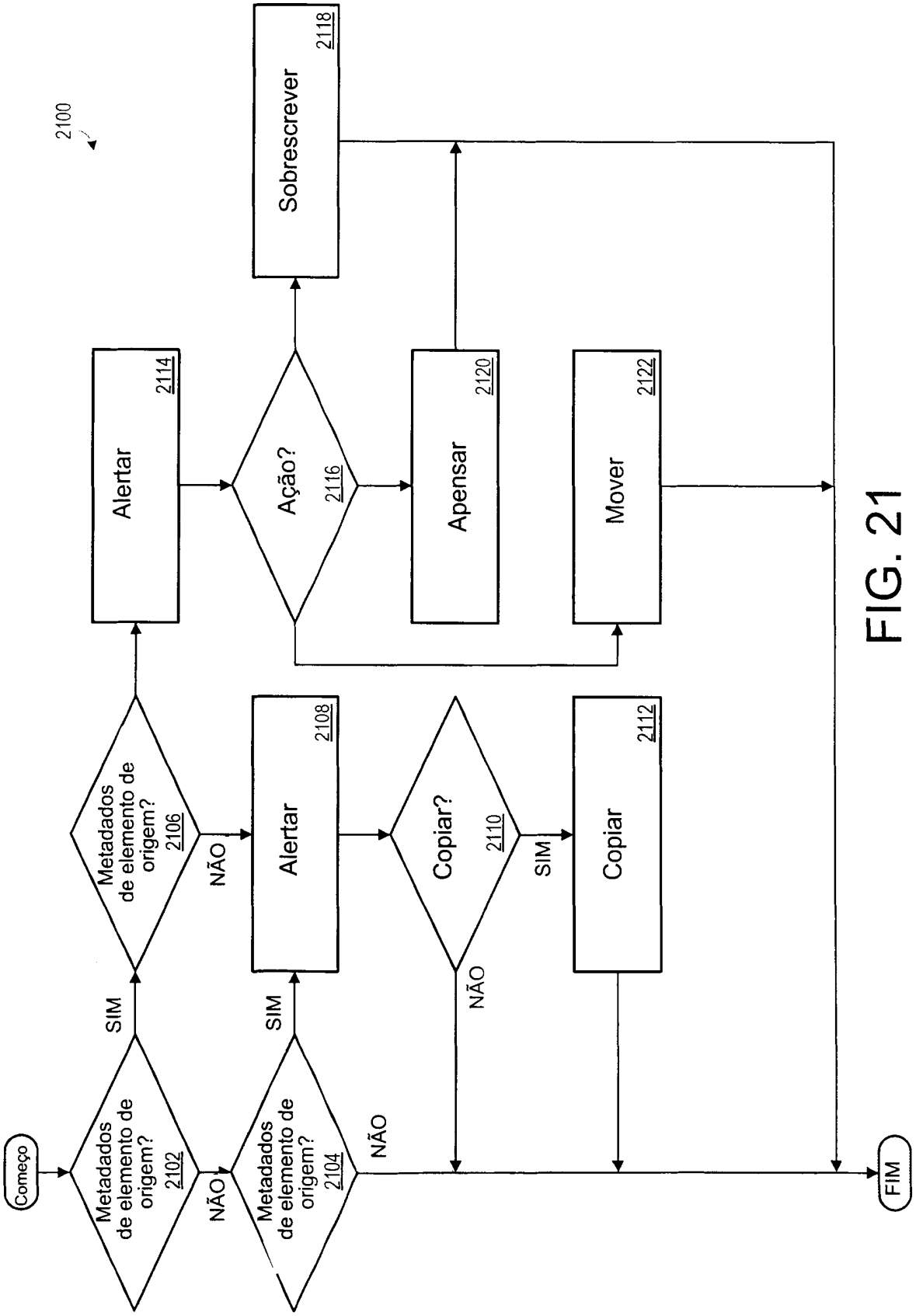

2100
Começo
Metadados de elemento de origem? 2102
SIM
NÃO
Metadados de elemento de origem? 2104
SIM
NÃO
Metadados de elemento de origem? 2106
NÃO
Alertar 2108
Copiar? 2110
SIM
NÃO
Copiar 2112
Alertar 2114
Ação? 2116
Sobrescrever 2118
Apensar 2120
Mover 2122
FIM

FIG. 21

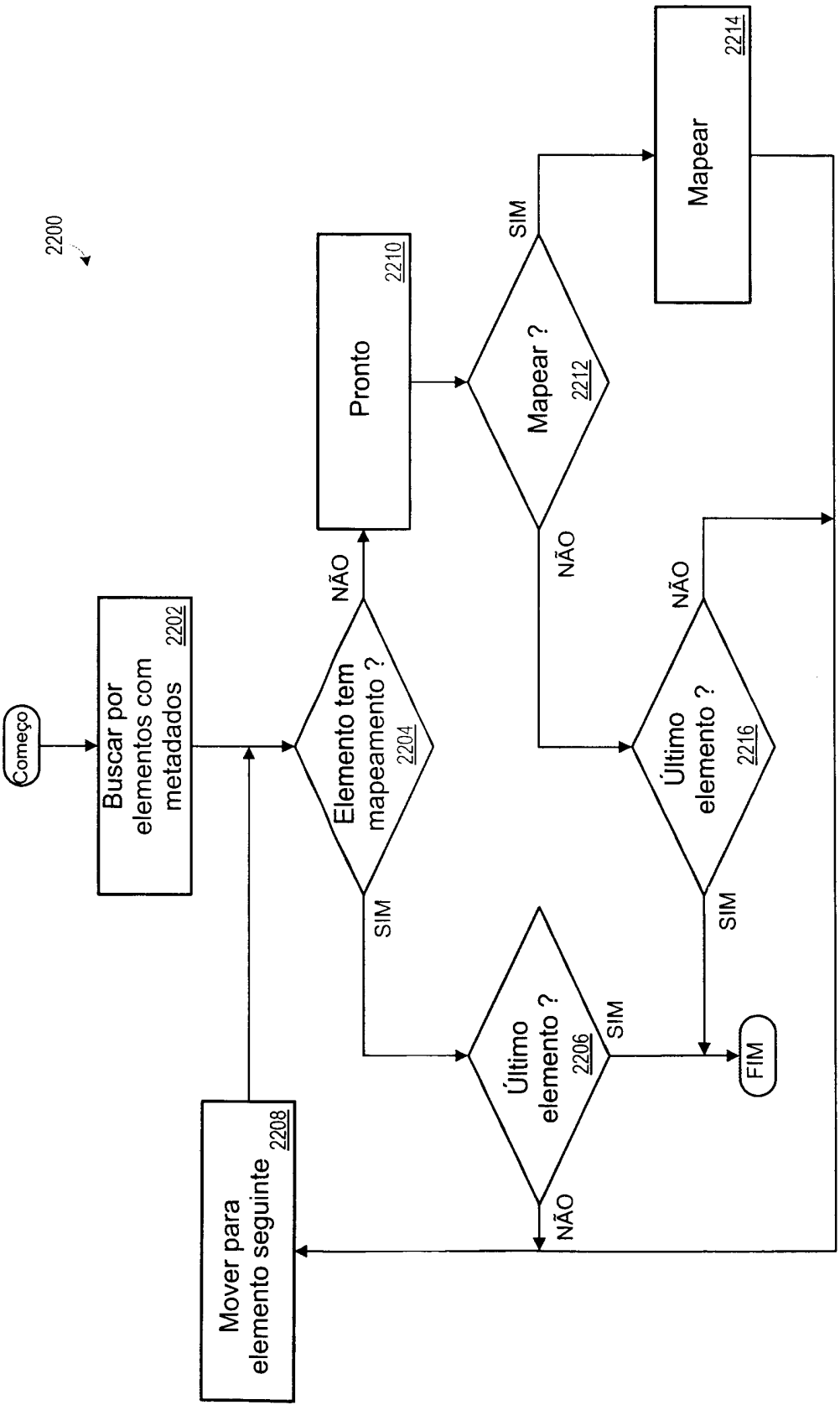

2200
Começo
Buscar por elementos com metadados 2202
Elemento tem mapeamento ? 2204
NÃO
SIM
Pronto 2210
Mapear ? 2212
SIM
NÃO
Mapear 2214
Último elemento ? 2206
SIM
NÃO
Último elemento ? 2216
NÃO
SIM
FIM
Mover para elemento seguinte 2208

FIG. 22

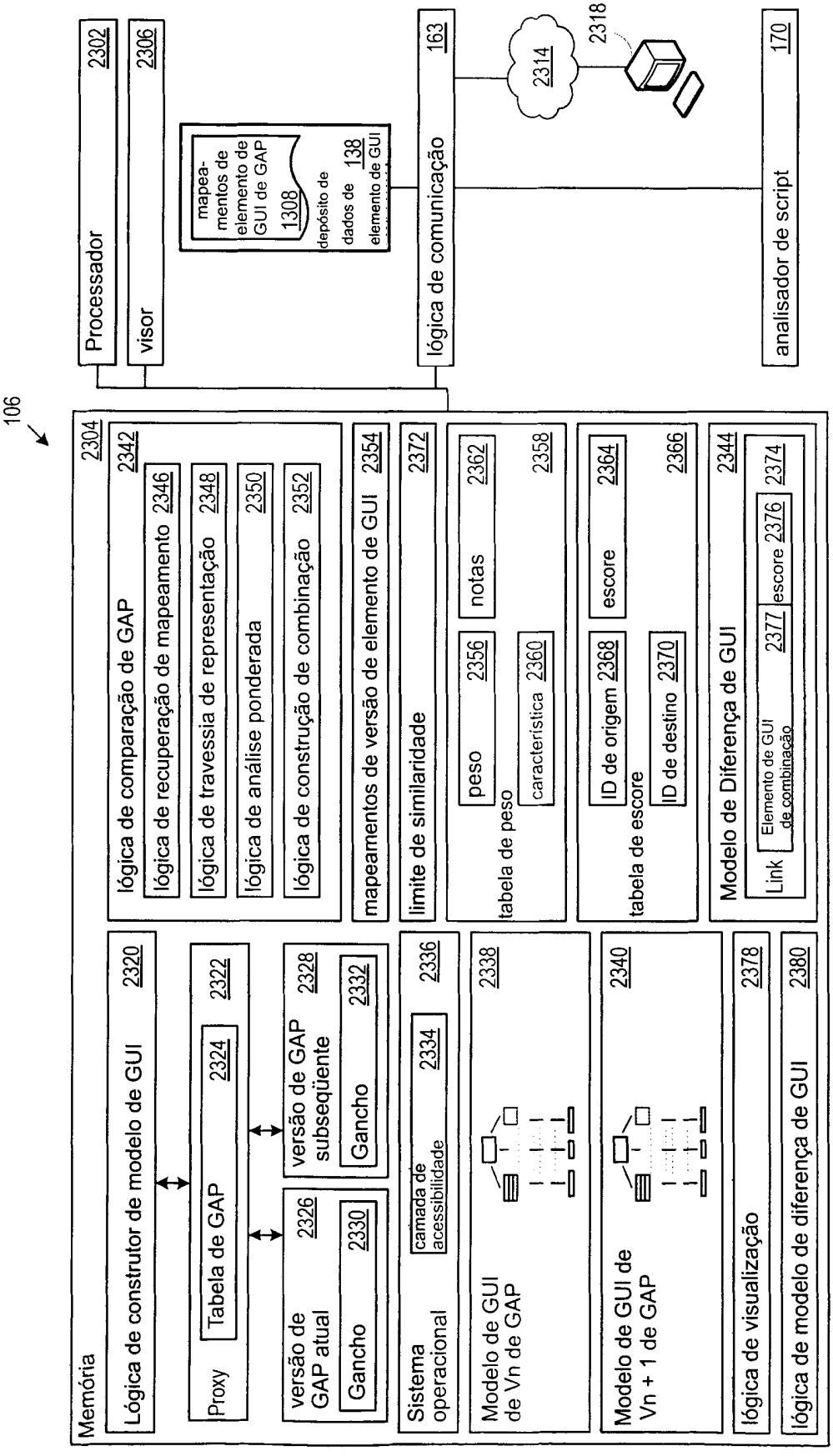

106
Memória
2304
Lógica de construtor de modelo de GUI
2320
Proxy
Tabela de GAP
2324
2322
versão de GAP atual
2326
Gancho
2330
versão de GAP subseqüente
2328
Gancho
2332
Sistema operacional
camada de acessibilidade
2334
2336
Modelo de GUI de Vn de GAP
2338
Modelo de GUI de Vn + 1 de GAP
2340
lógica de visualização
2378
lógica de modelo de diferença de GUI
2380
lógica de comparação de GAP
2342
lógica de recuperação de mapeamento
2346
lógica de travessia de representação
2348
lógica de análise ponderada
2350
lógica de construção de combinação
2352
mapeamentos de versão de elemento de GUI
2354
limite de similaridade
2372
tabela de peso
peso
2356
característica
2360
notas
2362
2358
tabela de escore
ID de origem
2368
ID de destino
2370
escore
2364
2366
Modelo de Diferença de GUI
2344
Link
Elemento de GUI de combinação
2377
escore
2376
2374
Processador
2302
visor
2306
mapeamentos de elemento de GUI de GAP
1308
depósito de dados de elemento de GUI
138
lógica de comunicação
163
2314
2318
analisador de script
170

FIG. 23

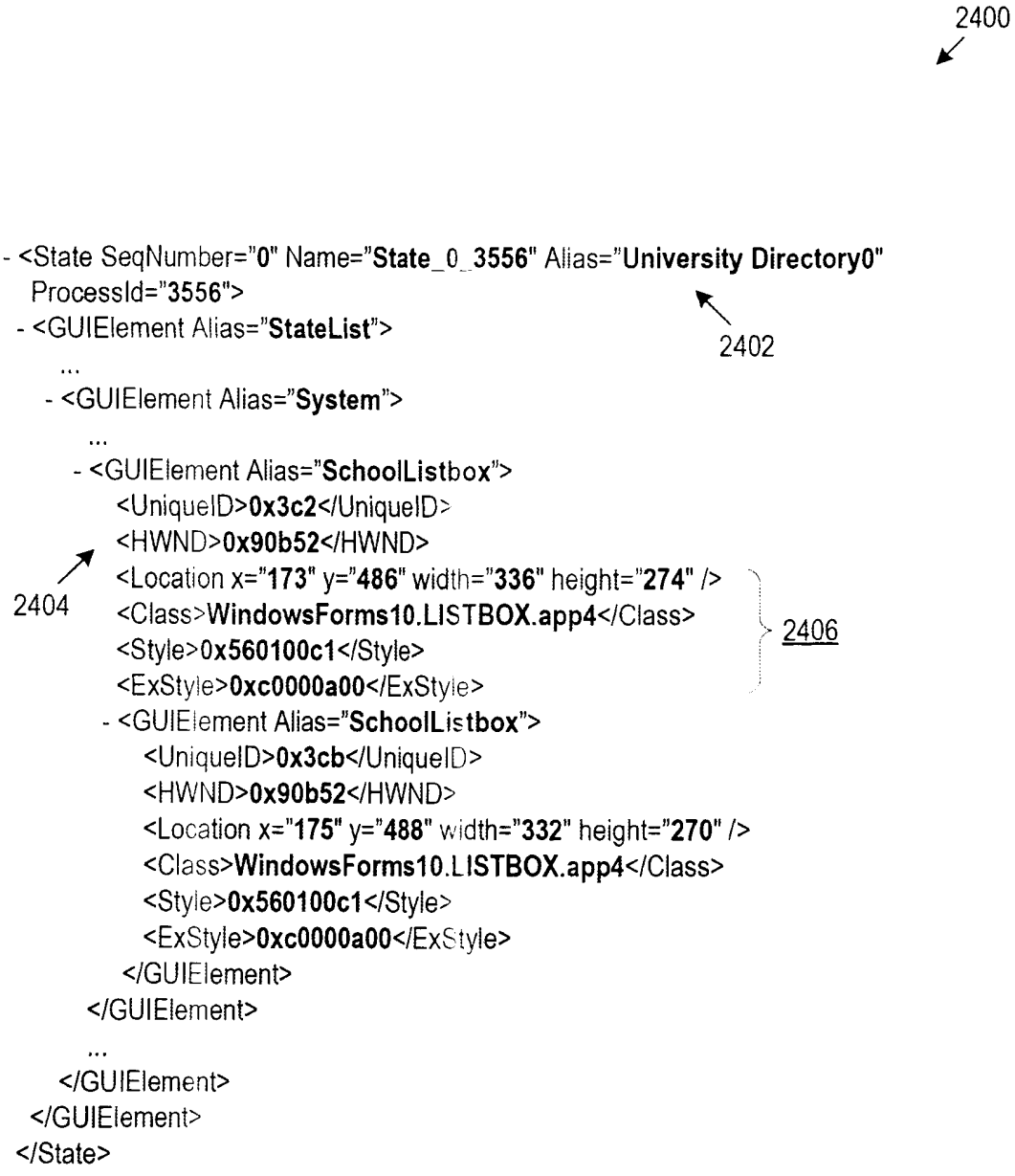

2400
- <State SeqNumber="0" Name="State_0_3556" Alias="University Directory0"
ProcessId="3556">
2402
- <GUIElement Alias="StateList">
...
- <GUIElement Alias="System">
...
- <GUIElement Alias="SchoolListbox">
<UniqueID>0x3c2</UniqueID>
<HWND>0x90b52</HWND>
2404
<Location x="173" y="486" width="336" height="274" />
<Class>WindowsForms10.LISTBOX.app4</Class>
<Style>0x560100c1</Style>
<ExStyle>0xc0000a00</ExStyle>
2406
- <GUIElement Alias="SchoolListbox">
<UniqueID>0x3cb</UniqueID>
<HWND>0x90b52</HWND>
<Location x="175" y="488" width="332" height="270" />
<Class>WindowsForms10.LISTBOX.app4</Class>
<Style>0x560100c1</Style>
<ExStyle>0xc0000a00</ExStyle>
</GUIElement>
</GUIElement>
...
</GUIElement>
</GUIElement>
</State>

FIG. 24


2500
- <State SeqNumber="0" Name="State_0_3068" Alias="University Directory1"
ProcessId="3068">
- <GUIElement Alias="StateList">
2502
...
- <GUIElement Alias="System">
...
- <GUIElement Alias="SchoolCombobox">
<UniqueID>0xa8</UniqueID>
<HWND>0x90e66</HWND>
2504
<Location x="248" y="653" width="299" height="24" />
<Class>WindowsForms10.COMBOBOX.app.0.378734a</Class>
<Style>0x560100242</Style>
<ExStyle>0xc0000800</ExStyle>
2506
- <GUIElement Alias="SchoolCombobox">
<UniqueID>0xb1</UniqueID>
<HWND>0x90e66</HWND>
<Location x="248" y="653" width="299" height="24" />
<Class>WindowsForms10.COMBOBOX.app.0.378734a</Class>
<Style>0x560100242</Style>
<ExStyle>0xc0000800</ExStyle>
- <GUIElement Alias="SchoolCombobox">
<UniqueID>0xb2</UniqueID>
<HWND>0x80e94</HWND>
<Location x="251" y="656" width="276" height="18" />
<Class>Edit</Class>
<Style>0x50000380</Style>
<ExStyle>0xc0000800</ExStyle>
...
</GUIElement>
</GUIElement>
</GUIElement>
...
</GUIElement>
</GUIElement>
</State>

FIG. 25

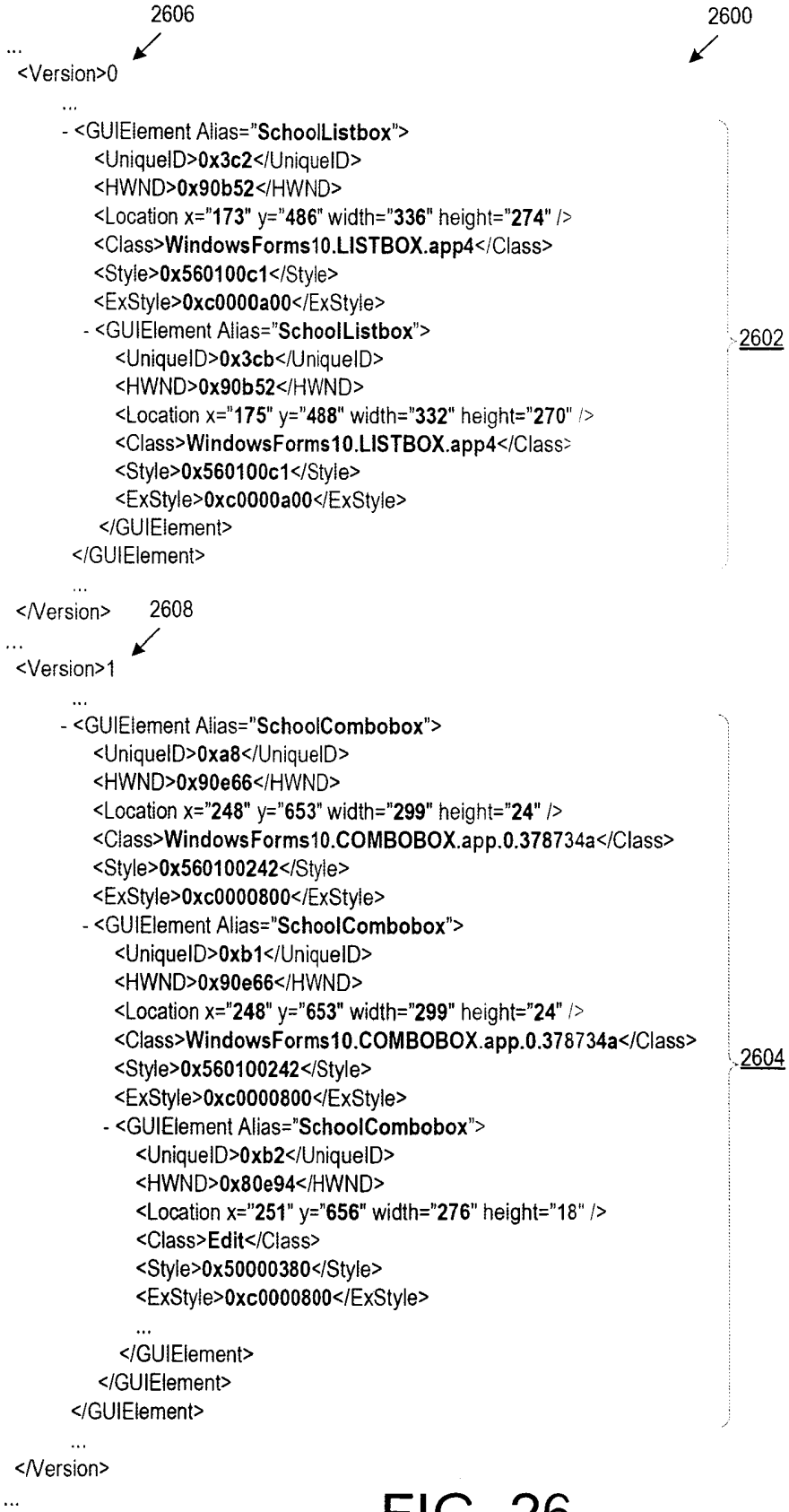
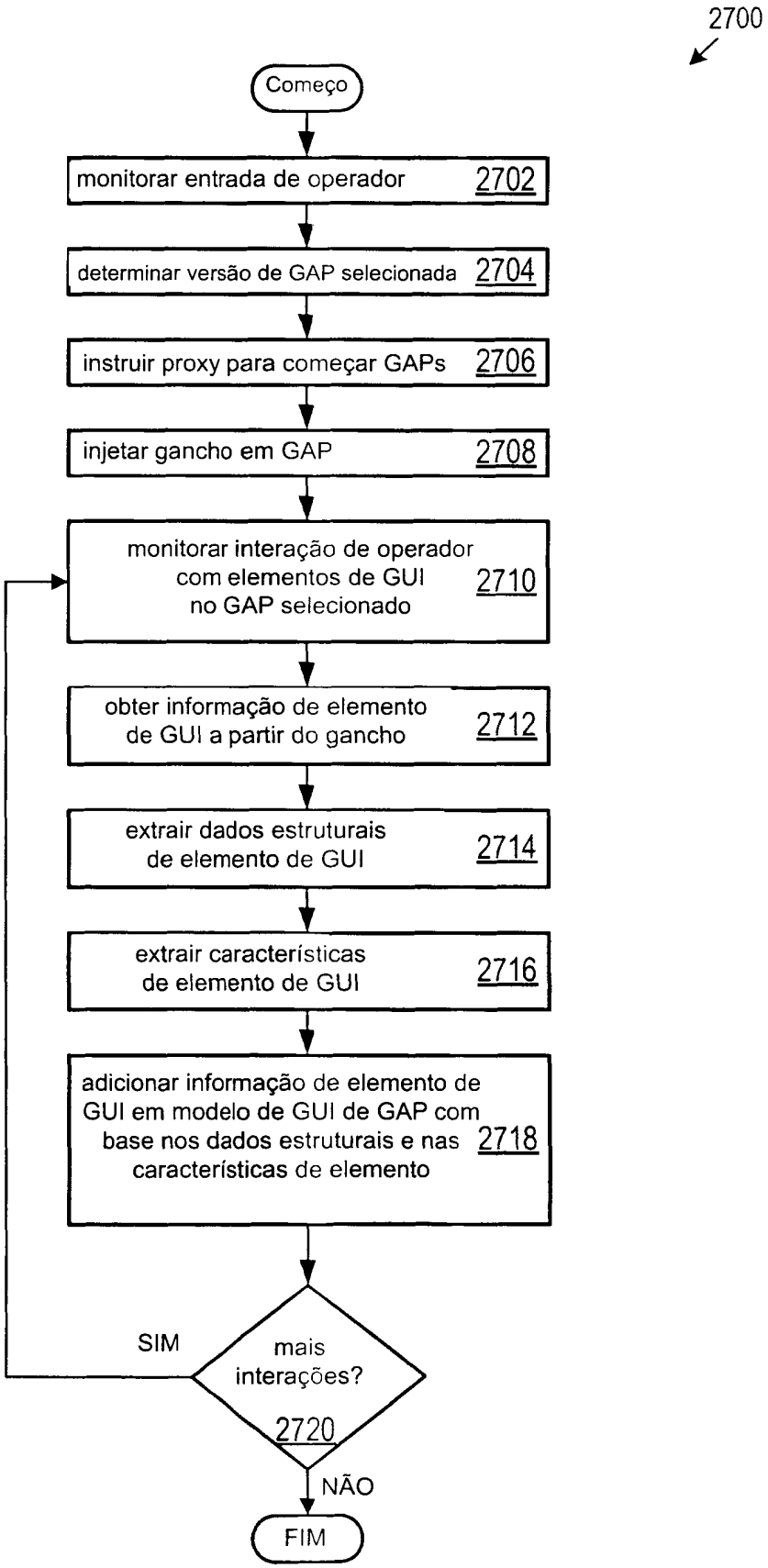

2606
2600
...
<Version>0
...
- <GUIElement Alias="SchoolListbox">
<UniqueID>0x3c2</UniqueID>
<HWND>0x90b52</HWND>
<Location x="173" y="486" width="336" height="274" />
<Class>WindowsForms10.LISTBOX.app4</Class>
<Style>0x560100c1</Style>
<ExStyle>0xc0000a00</ExStyle>
- <GUIElement Alias="SchoolListbox">
<UniqueID>0x3cb</UniqueID>
<HWND>0x90b52</HWND>
<Location x="175" y="488" width="332" height="270" />
<Class>WindowsForms10.LISTBOX.app4</Class>
<Style>0x560100c1</Style>
<ExStyle>0xc0000a00</ExStyle>
</GUIElement>
</GUIElement>
...
2602
</Version>
2608
...
<Version>1
...
- <GUIElement Alias="SchoolCombobox">
<UniqueID>0xa8</UniqueID>
<HWND>0x90e66</HWND>
<Location x="248" y="653" width="299" height="24" />
<Class>WindowsForms10.COMBOBOX.app.0.378734a</Class>
<Style>0x560100242</Style>
<ExStyle>0xc0000800</ExStyle>
- <GUIElement Alias="SchoolCombobox">
<UniqueID>0xb1</UniqueID>
<HWND>0x90e66</HWND>
<Location x="248" y="653" width="299" height="24" />
<Class>WindowsForms10.COMBOBOX.app.0.378734a</Class>
<Style>0x560100242</Style>
<ExStyle>0xc0000800</ExStyle>
- <GUIElement Alias="SchoolCombobox">
<UniqueID>0xb2</UniqueID>
<HWND>0x80e94</HWND>
<Location x="251" y="656" width="276" height="18" />
<Class>Edit</Class>
<Style>0x50000380</Style>
<ExStyle>0xc0000800</ExStyle>
...
</GUIElement>
</GUIElement>
</GUIElement>
...
2604
</Version>
...

FIG. 26


2700
Começo
monitorar entrada de operador 2702
determinar versão de GAP selecionada 2704
instruir proxy para começar GAPs 2706
injetar gancho em GAP 2708
monitorar interação de operador com elementos de GUI no GAP selecionado 2710
obter informação de elemento de GUI a partir do gancho 2712
extrair dados estruturais de elemento de GUI 2714
extrair características de elemento de GUI 2716
adicionar informação de elemento de GUI em modelo de GUI de GAP com base nos dados estruturais e nas características de elemento 2718
mais interações? 2720
SIM
NÃO
FIM

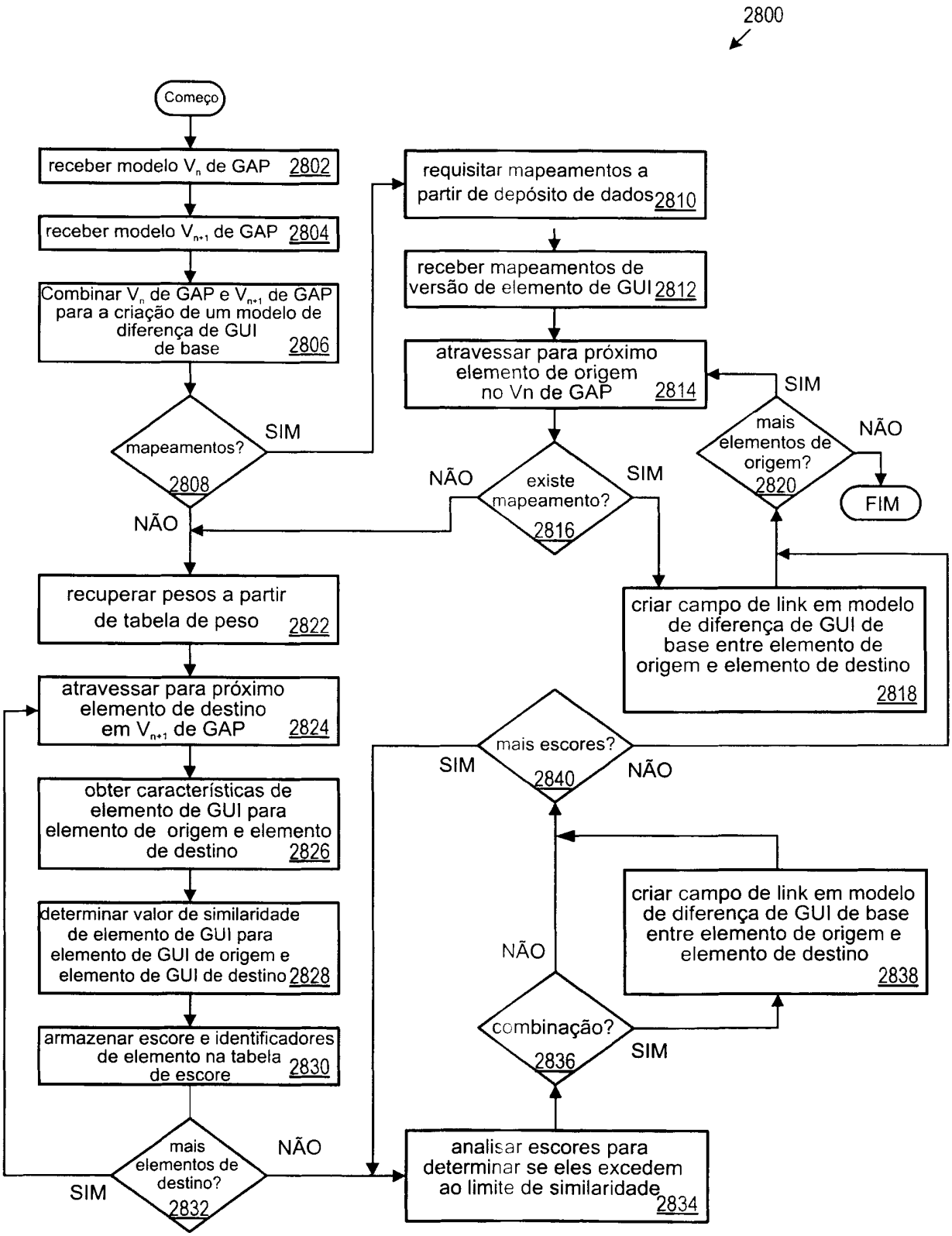

2800
Começo
receber modelo V_n de GAP 2802
receber modelo V_{n+1} de GAP 2804
Combinar V_n de GAP e V_{n+1} de GAP para a criação de um modelo de diferença de GUI de base 2806
mapeamentos? 2808
SIM
NÃO
requisitar mapeamentos a partir de depósito de dados 2810
receber mapeamentos de versão de elemento de GUI 2812
atravessar para próximo elemento de origem no Vn de GAP 2814
existe mapeamento? 2816
NÃO
SIM
mais elementos de origem? 2820
SIM
NÃO
FIM
criar campo de link em modelo de diferença de GUI de base entre elemento de origem e elemento de destino 2818
recuperar pesos a partir de tabela de peso 2822
atravessar para próximo elemento de destino em V_{n+1} de GAP 2824
obter características de elemento de GUI para elemento de origem e elemento de destino 2826
determinar valor de similaridade de elemento de GUI para elemento de GUI de origem e elemento de GUI de destino 2828
armazenar escore e identificadores de elemento na tabela de escore 2830
mais elementos de destino? 2832
SIM
NÃO
mais escores? 2840
SIM
NÃO
criar campo de link em modelo de diferença de GUI de base entre elemento de origem e elemento de destino 2838
combinação? 2836
NÃO
SIM
analisar escores para determinar se eles excedem ao limite de similaridade 2834

FIG. 28

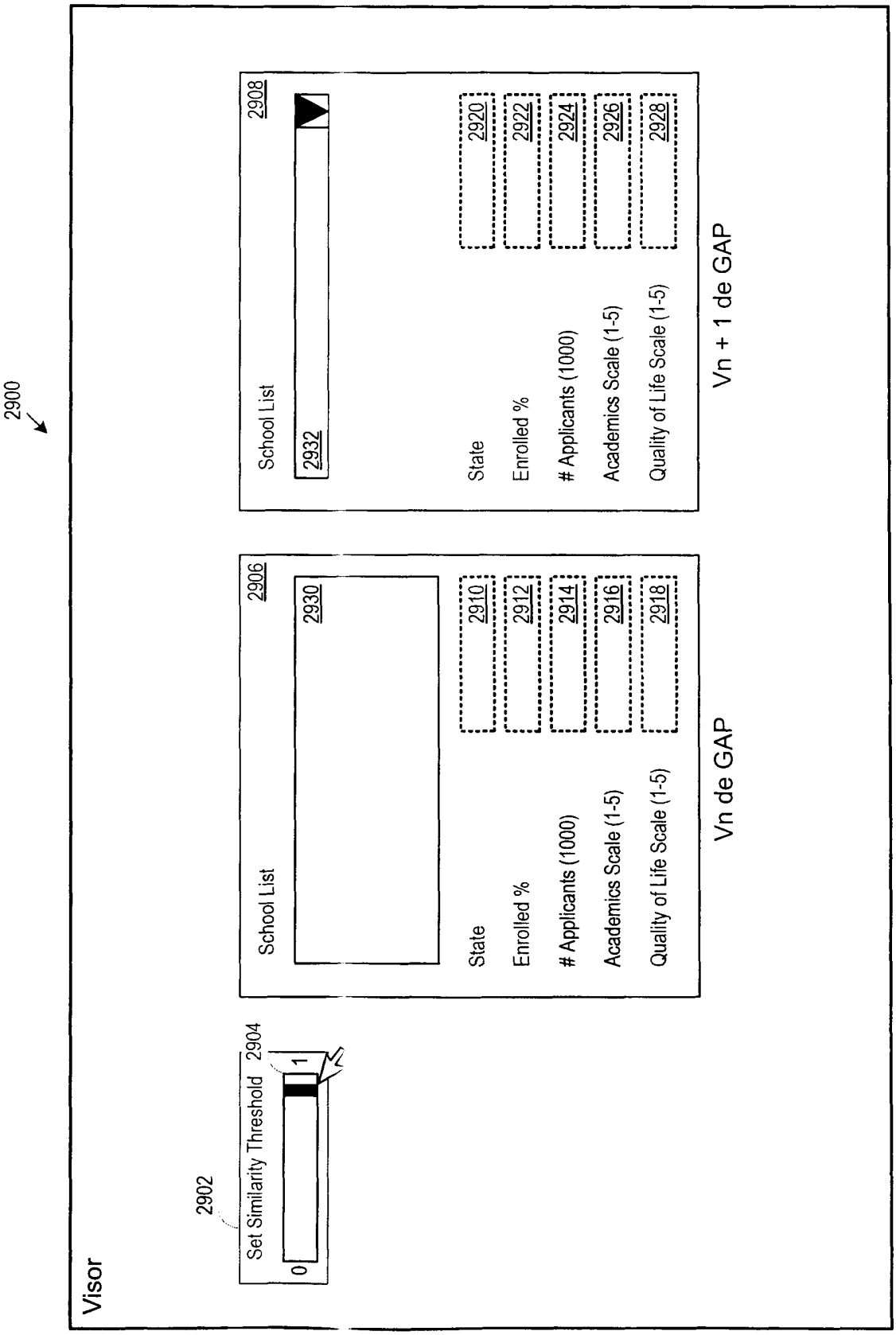

2900
Visor
2902
Set Similarity Threshold
2904
0
1
School List
2906
2930
State
2910
Enrolled %
2912
Applicants (1000)
2914
Academics Scale (1-5)
2916
Quality of Life Scale (1-5)
2918
Vn de GAP
School List
2908
2932
State
2920
Enrolled %
2922
Applicants (1000)
2924
Academics Scale (1-5)
2926
Quality of Life Scale (1-5)
2928
Vn + 1 de GAP

FIG. 29

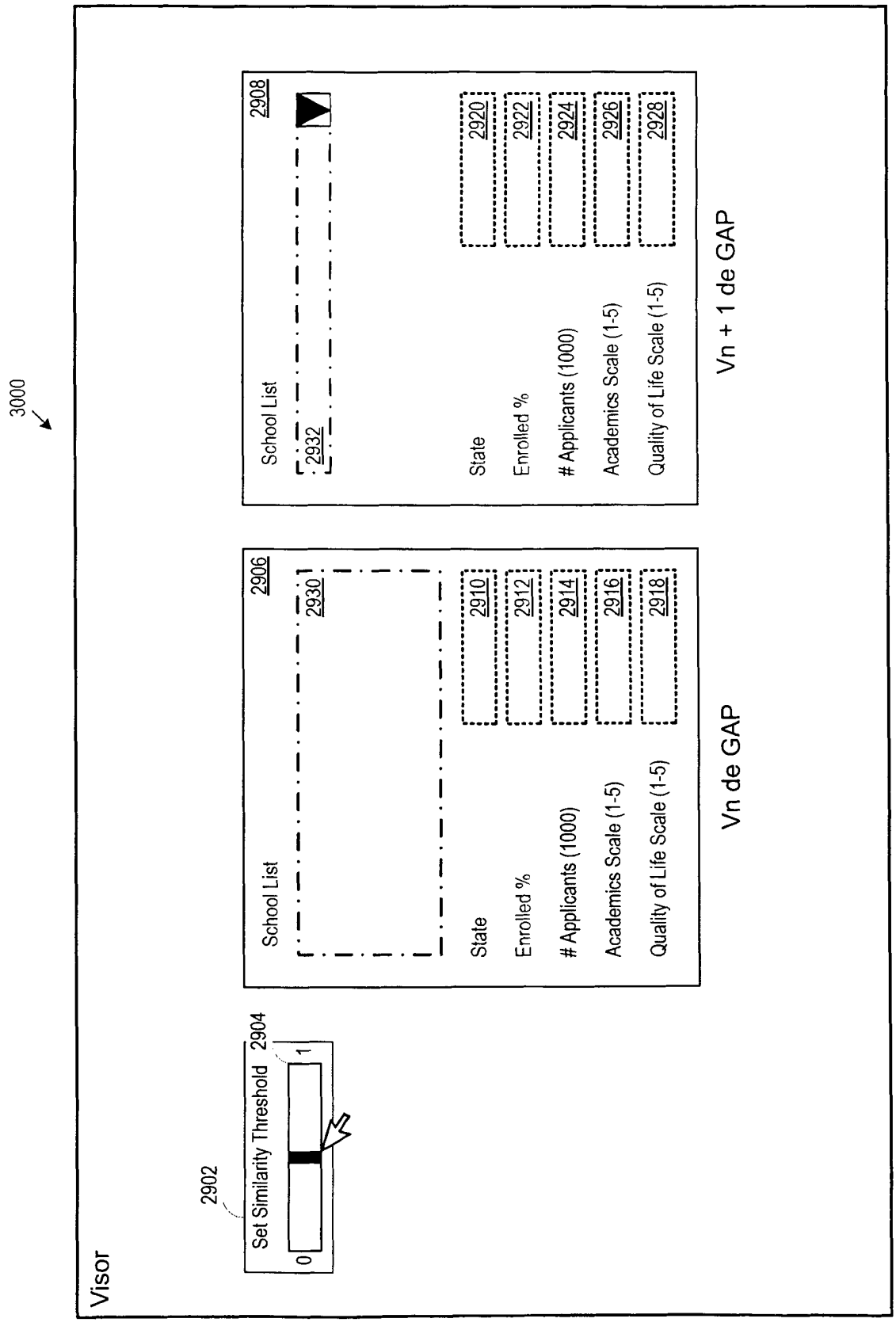

3000
Visor
2902
Set Similarity Threshold
2904
0
1
School List
2906
2930
State
2910
Enrolled %
2912
Applicants (1000)
2914
Academics Scale (1-5)
2916
Quality of Life Scale (1-5)
2918
Vn de GAP
School List
2908
2932
State
2920
Enrolled %
2922
Applicants (1000)
2924
Academics Scale (1-5)
2926
Quality of Life Scale (1-5)
2928
Vn + 1 de GAP

FIG. 30

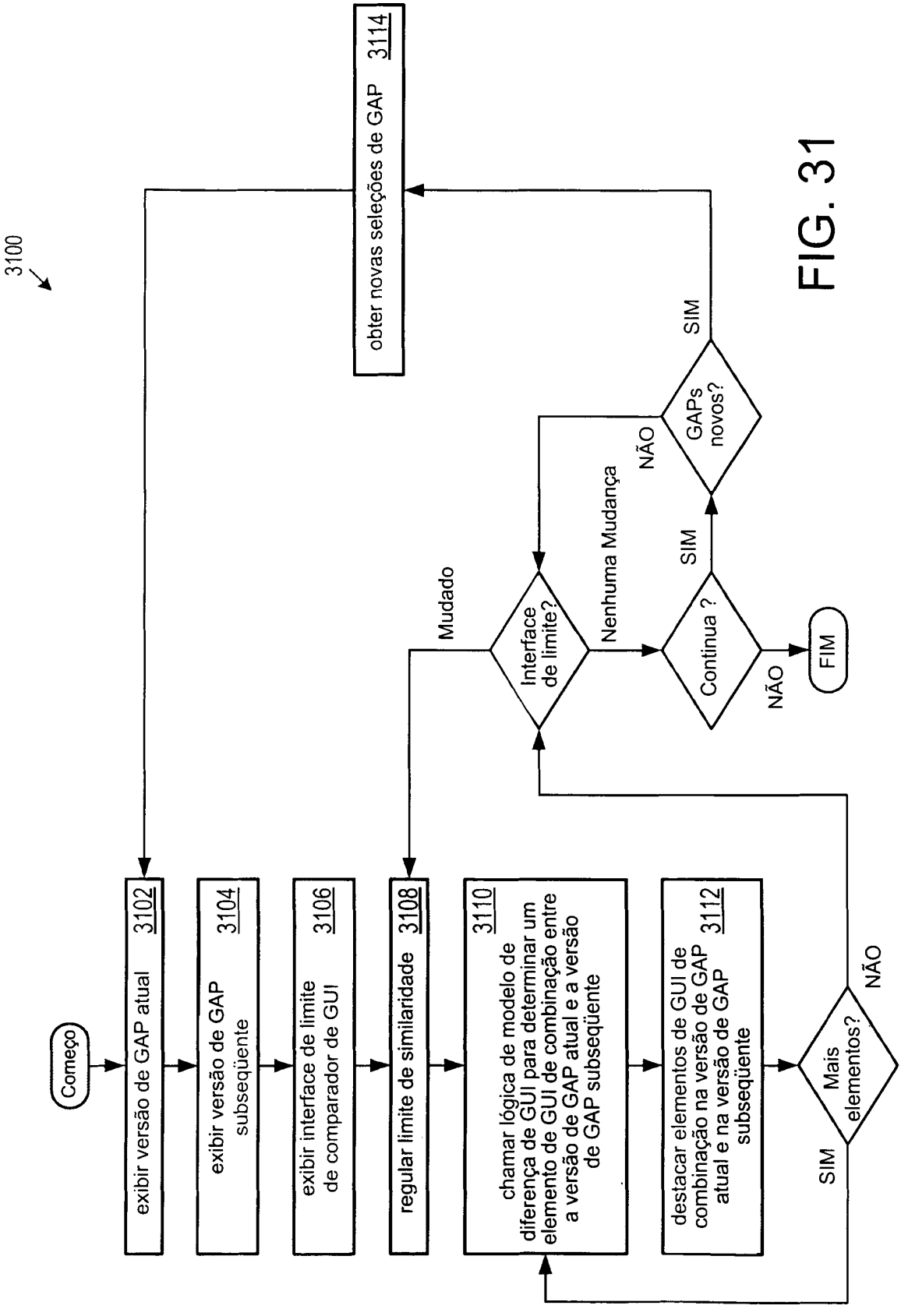

3100
Começo
exibir versão de GAP atual 3102
exibir versão de GAP subseqüente 3104
exibir interface de limite de comparador de GUI 3106
regular limite de similaridade 3108
chamar lógica de modelo de diferença de GUI para determinar um elemento de GUI de combinação entre a versão de GAP atual e a versão de GAP subseqüente 3110
destacar elementos de GUI de combinação na versão de GAP atual e na versão de GAP subseqüente 3112
Mais elementos?
SIM
NÃO
Interface de limite?
Mudado
Nenhuma Mudança
Continua ?
SIM
NÃO
FIM
GAPs novos?
SIM
NÃO
obter novas seleções de GAP 3114

FIG. 31

3200

3202

Property 1 if x is the root of g_1 and y is the root of g_2, then $x \sim y$;

3204

Property 2 if $x \sim y$ and one of x or y is the root node in its graph, then the other node is the root node as well;

3206

Property 3 if $x \sim y$ and `type(`x`)=type(`y`)`, and nodes x, y are not tagged as potentially deleted, and $x \xrightarrow{l_p^q} x'$ in g_1, then there exists an edge $y \xrightarrow{l_k^m} y'$ in g_2, with the same labels `r=s`, $r \neq \diamond$ and $s \neq \diamond$, and `max(r)=max(s)` and `min(r)=min(s)`, and `type(`x'`)=type(`y'`)`, such that $x' \sim y'$;

3208

Property 4 conversely, if $x \sim y$ and `type(`x`)=type(`y`)`, and nodes x, y are not tagged as potentially deleted, and $y \xrightarrow{l_k^m} y'$ in g_2, then there exists an edge $x \xrightarrow{l_p^q} x'$ in g_1, with the same label `l`, $l \neq \diamond$, and `p=k` and `m=q`, and `type(`x'`)=type(`y'`)`, such that $x' \sim y'$.

3210

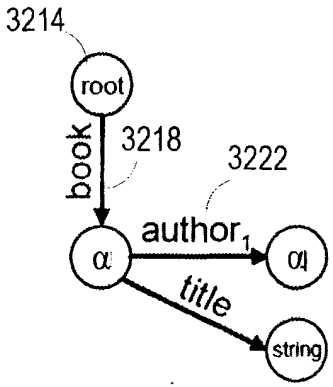
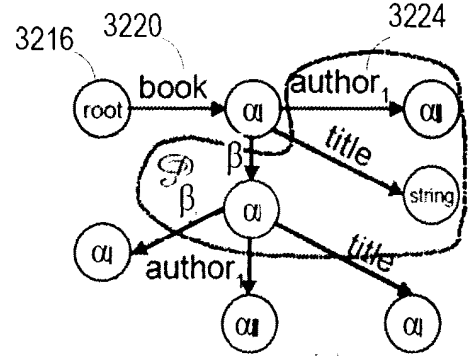


FIG. 32

3212



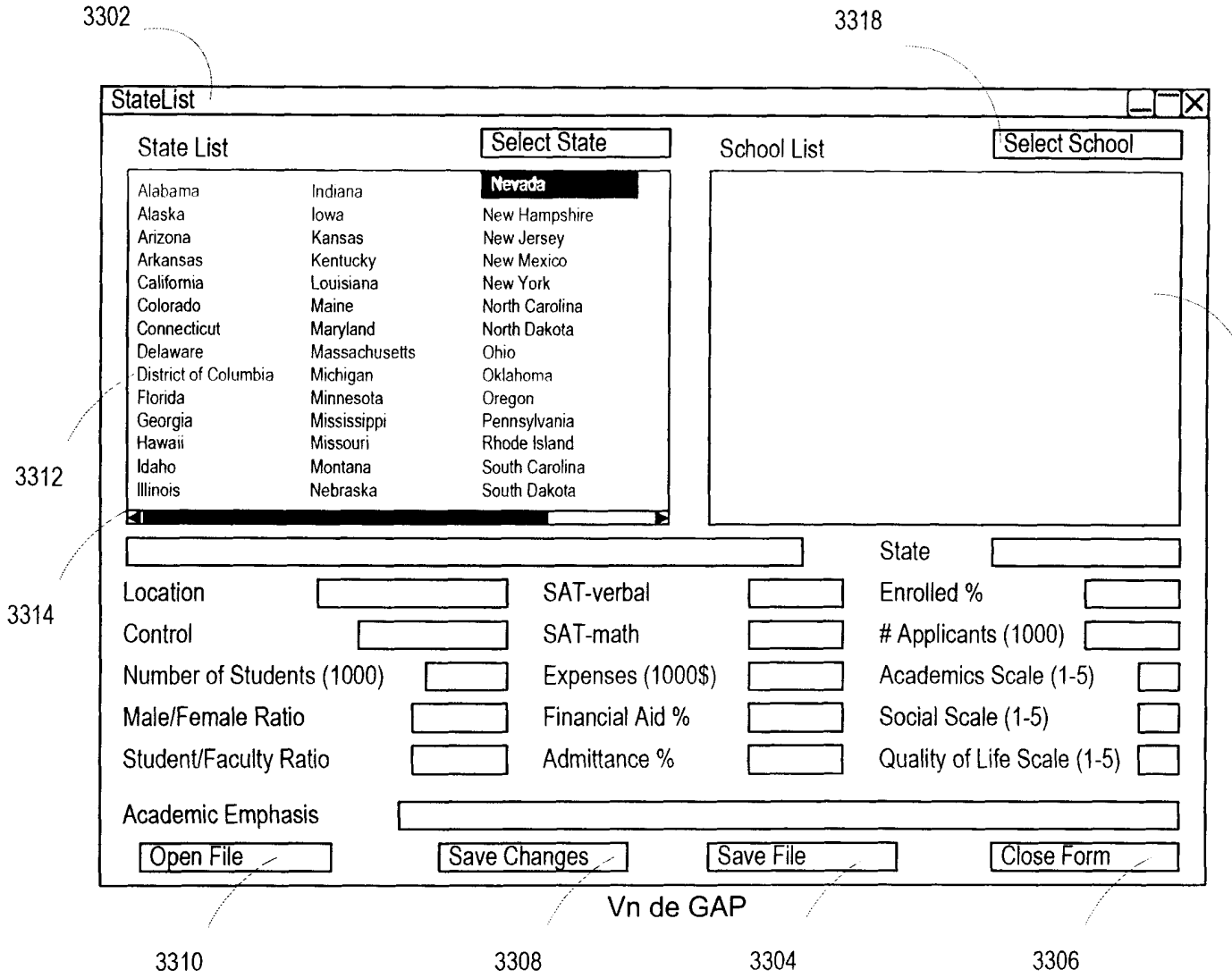

3302
3318
StateList
State List
Select State
School List
Select School
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
3316
3312
State
3314
Location
SAT-verbal
Enrolled %
Control
SAT-math
Applicants (1000)
Number of Students (1000)
Expenses (1000$)
Academics Scale (1-5)
Male/Female Ratio
Financial Aid %
Social Scale (1-5)
Student/Faculty Ratio
Admittance %
Quality of Life Scale (1-5)
Academic Emphasis
Open File
Save Changes
Save File
Close Form
Vn de GAP
3310
3308
3304
3306
150

FIG. 33

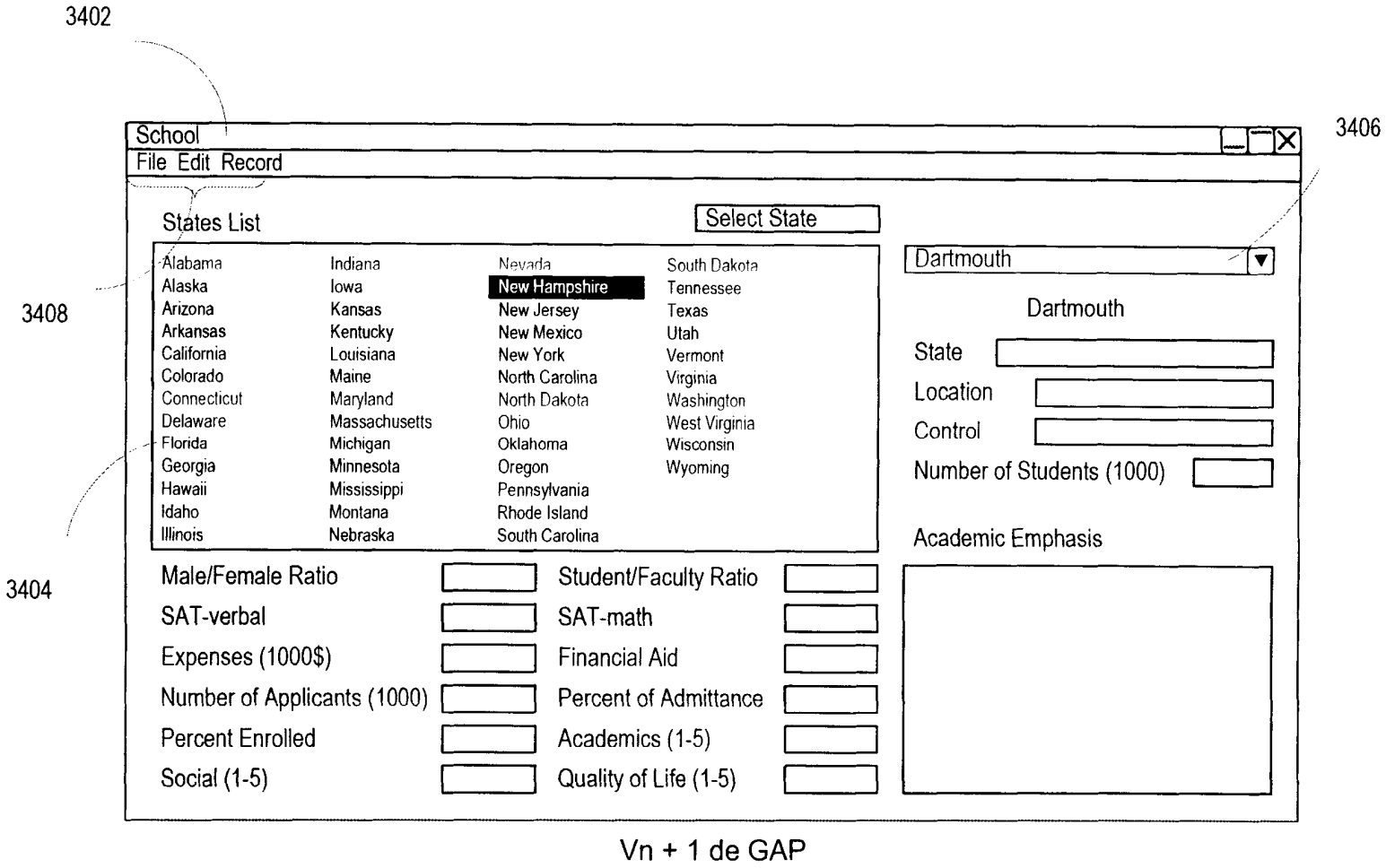

3402
3406
3408
3404
School
File Edit Record
States List
Select State
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
Dartmouth
Dartmouth
State
Location
Control
Number of Students (1000)
Academic Emphasis
Male/Female Ratio
Student/Faculty Ratio
SAT-verbal
SAT-math
Expenses (1000$)
Financial Aid
Number of Applicants (1000)
Percent of Admittance
Percent Enrolled
Academics (1-5)
Social (1-5)
Quality of Life (1-5)
Vn + 1 de GAP

FIG. 34

152

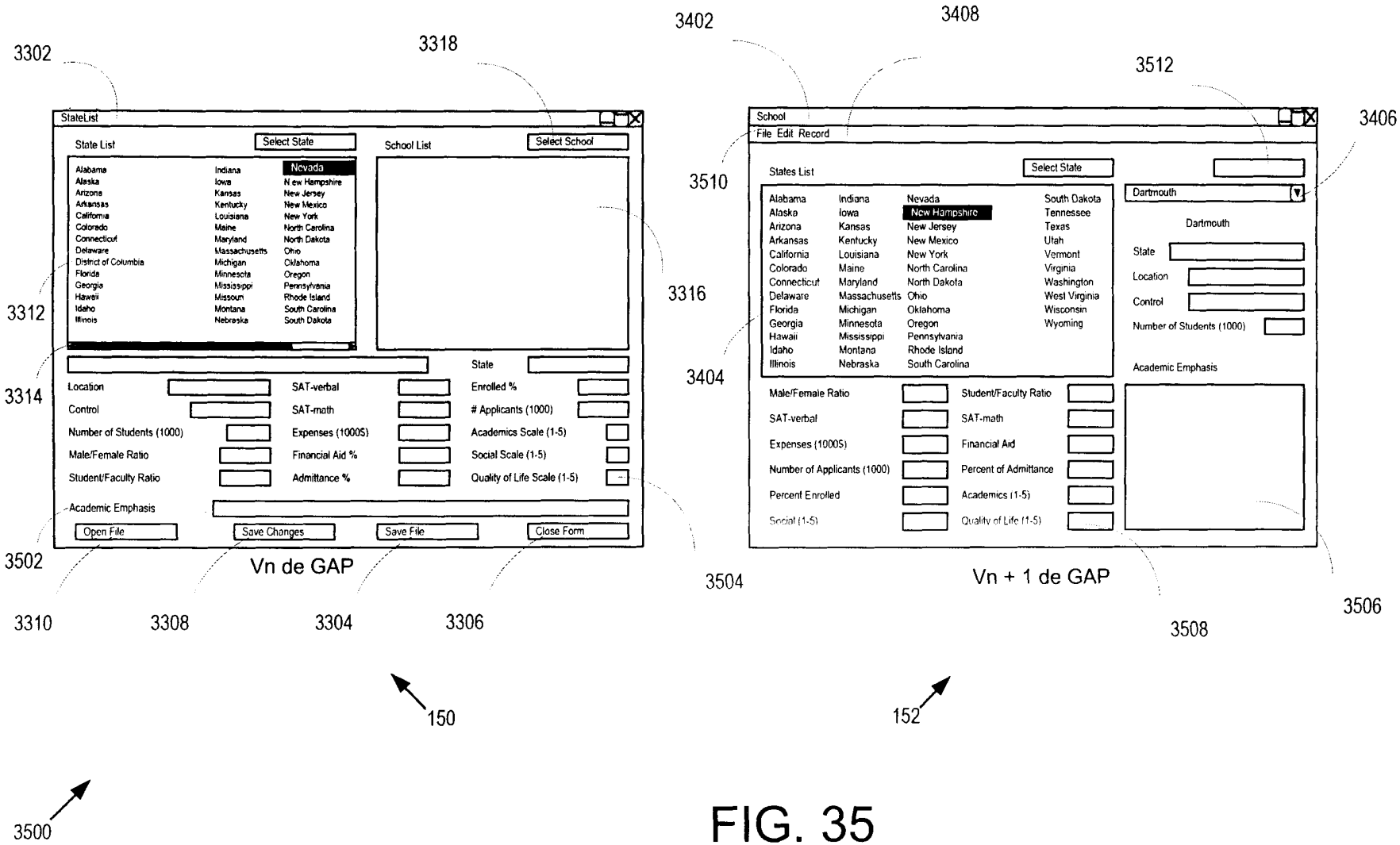

3302
3318
StateList
State List
Select State
School List
Select School
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
3312
3314
State
Location
SAT-verbal
Enrolled %
Control
SAT-math
Applicants (1000)
Number of Students (1000)
Expenses (1000$)
Academics Scale (1-5)
Male/Female Ratio
Financial Aid %
Social Scale (1-5)
Student/Faculty Ratio
Admittance %
Quality of Life Scale (1-5)
Academic Emphasis
Open File
Save Changes
Save File
Close Form
3502
Vn de GAP
3310
3308
3304
3306
150
3402
3408
3512
3406
3510
School
File Edit Record
States List
Select State
Dartmouth
Dartmouth
State
Location
Control
Number of Students (1000)
Academic Emphasis
3316
3404
Male/Female Ratio
Student/Faculty Ratio
SAT-verbal
SAT-math
Expenses (1000$)
Financial Aid
Number of Applicants (1000)
Percent of Admittance
Percent Enrolled
Academics (1-5)
Social (1-5)
Quality of Life (1-5)
3504
Vn + 1 de GAP
3506
3508
152
3500

FIG. 35

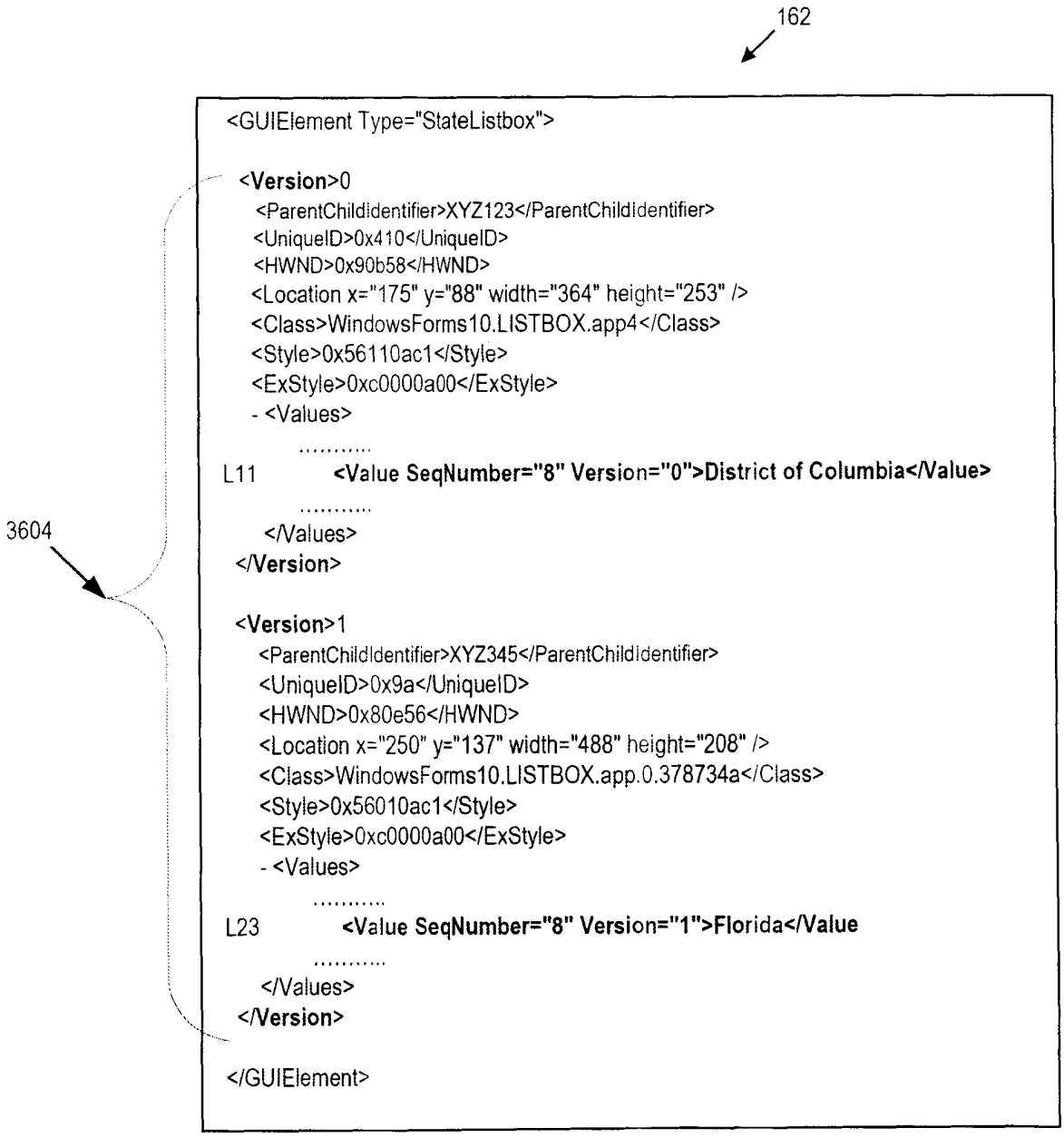

162
<GUIElement Type="StateListbox">
<Version>0
<ParentChildIdentifier>XYZ123</ParentChildIdentifier>
<UniqueID>0x410</UniqueID>
<HWND>0x90b58</HWND>
<Location x="175" y="88" width="364" height="253" />
<Class>WindowsForms10.LISTBOX.app4</Class>
<Style>0x56110ac1</Style>
<ExStyle>0xc0000a00</ExStyle>
- <Values>
.....
L11 <Value SeqNumber="8" Version="0">District of Columbia</Value>
.....
</Values>
</Version>
3604
<Version>1
<ParentChildIdentifier>XYZ345</ParentChildIdentifier>
<UniqueID>0x9a</UniqueID>
<HWND>0x80e56</HWND>
<Location x="250" y="137" width="488" height="208" />
<Class>WindowsForms10.LISTBOX.app.0.378734a</Class>
<Style>0x56010ac1</Style>
<ExStyle>0xc0000a00</ExStyle>
- <Values>
.....
L23 <Value SeqNumber="8" Version="1">Florida</Value
.....
</Values>
</Version>
</GUIElement>

FIG. 36

```
'Testar script quanto a GUI de GAP atual quanto a abrir um arquivo, selecionar uma escola (Acme State
 University), mudar a escala acadêmica de 2 para 3 e salvar em um novo arquivo                    164

 L1   Window("StateList").WinObject("Open File").Click 86,12
 L2   Window("StateList").Dialog("Open").WinListView("SysListView32").Select "university.data"
 L3   Window("StateList").Dialog("Open").WinButton("Open").Click
 L4   Window("StateList").WinObject("StateListbox").Click 31,7
 L5   Window("StateList").WinObject("Select State").Click 36,14
 L6   Window("StateList").WinObject("SchoolListbox").Click 19,22
 L7   Window("StateList").WinObject("Select School").Click 67,12
 L8   Window("StateList").WinObject("AcadScale").Drag 18,14
 L9   Window("StateList").Drop 698,471
 L10  Window("StateList").WinObject("AcadScale").Type "3"
 L11  Window("StateList").WinObject("Save Change").Click 108,7
 L12  Window("StateList").WinObject("Save File").Click 70,16
 L13  Window("StateList").Dialog("Save a Data Record").WinEdit("File name:").Set “university_revise.data"
 L14  Window("StateList").Dialog("Save a Data Record").WinButton("Save").Click
```

FIG. 37

164

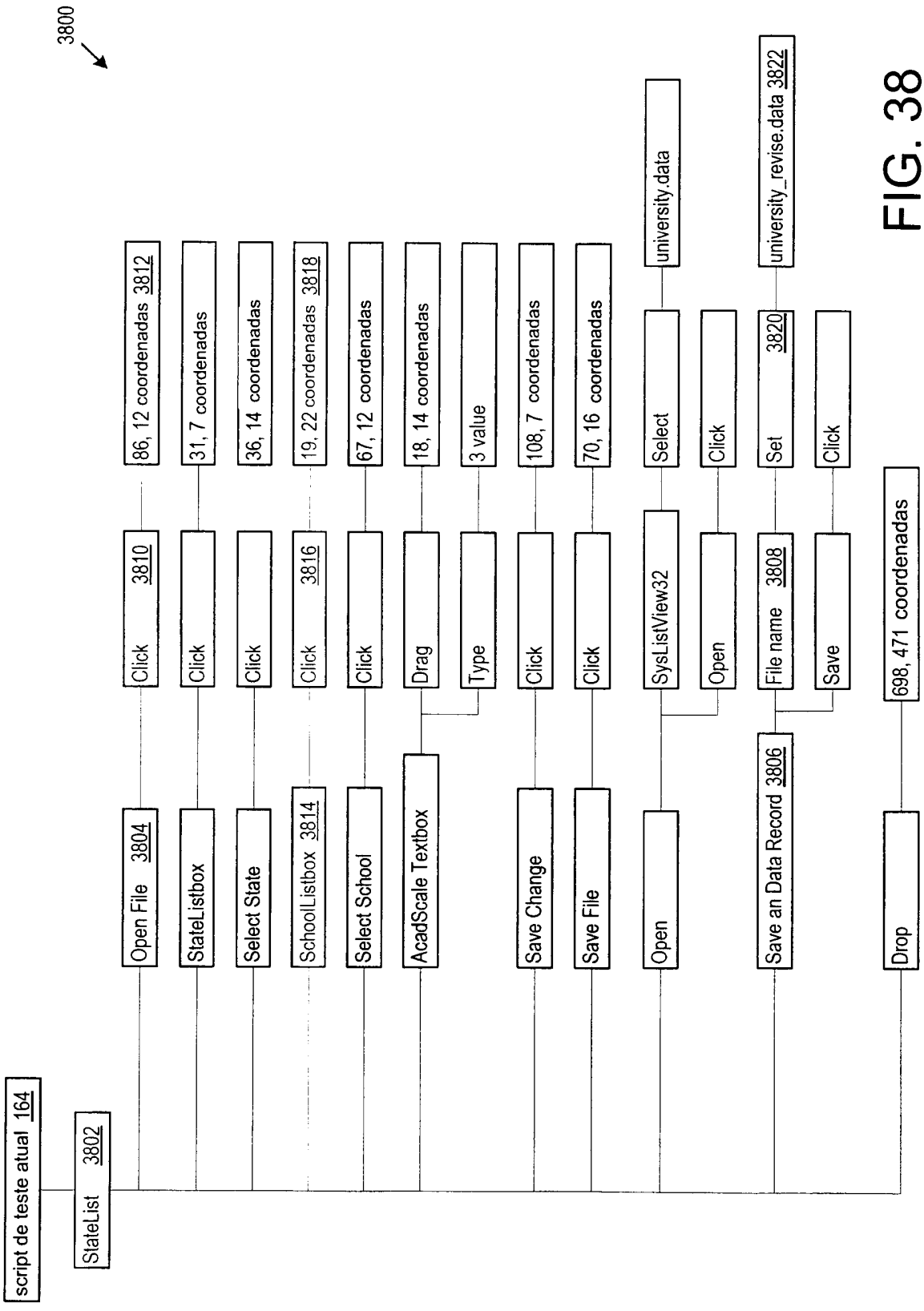

3800
script de teste atual 164
StateList 3802
Open File 3804
StateListbox
Select State
SchoolListbox 3814
Select School
AcadScale Textbox
Save Change
Save File
Open
Save an Data Record 3806
Drop
Click 3810
Click
Click
Click 3816
Click
Drag
Type
Click
Click
SysListView32
Open
File name 3808
Save
86, 12 coordenadas 3812
31, 7 coordenadas
36, 14 coordenadas
19, 22 coordenadas 3818
67, 12 coordenadas
18, 14 coordenadas
3 value
108, 7 coordenadas
70, 16 coordenadas
Select
Click
Set 3820
Click
698, 471 coordenadas
university.data
university_revise.data 3822

FIG. 38

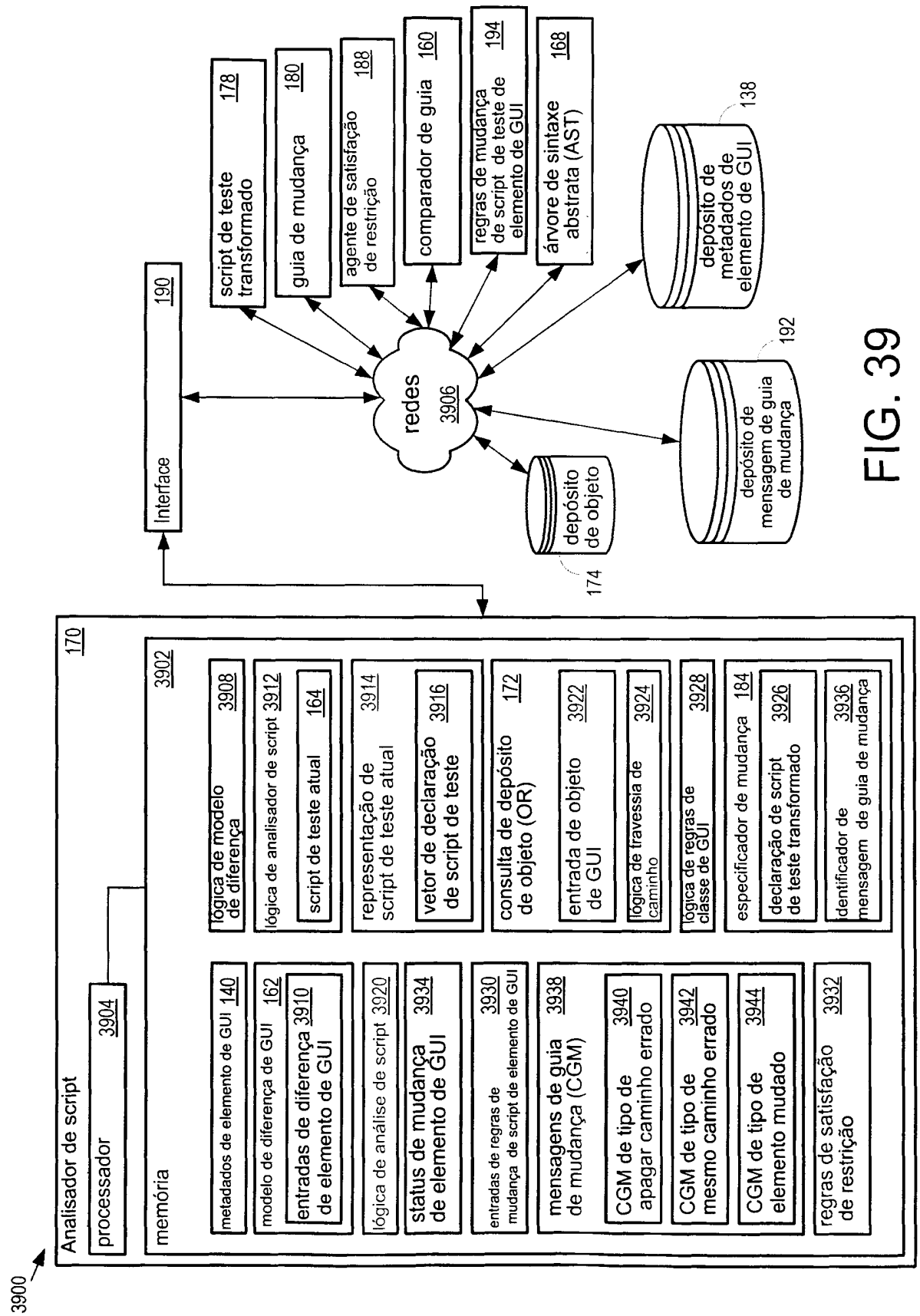

3900
Analisador de script
170
processador
3904
memória
3902
metadados de elemento de GUI 140
modelo de diferença de GUI 162
entradas de diferença de elemento de GUI 3910
lógica de análise de script 3920
status de mudança de elemento de GUI 3934
entradas de regras de mudança de script de elemento de GUI 3930
mensagens de guia de mudança (CGM) 3938
CGM de tipo de apagar caminho errado 3940
CGM de tipo de mesmo caminho errado 3942
CGM de tipo de elemento mudado 3944
regras de satisfação de restrição 3932
lógica de modelo de diferença 3908
lógica de analisador de script 3912
script de teste atual 164
representação de script de teste atual 3914
vetor de declaração de script de teste 3916
consulta de depósito de objeto (OR) 172
entrada de objeto de GUI 3922
lógica de travessia de caminho 3924
lógica de regras de classe de GUI 3928
especificador de mudança 184
declaração de script de teste transformado 3926
identificador de mensagem de guia de mudança 3936
Interface 190
redes 3906
script de teste transformado 178
guia de mudança 180
agente de satisfação de restrição 188
comparador de guia 160
regras de mudança de script de teste de elemento de GUI 194
árvore de sintaxe abstrata (AST) 168
depósito de metadados de elemento de GUI 138
depósito de mensagem de guia de mudança 192
depósito de objeto 174

FIG. 39

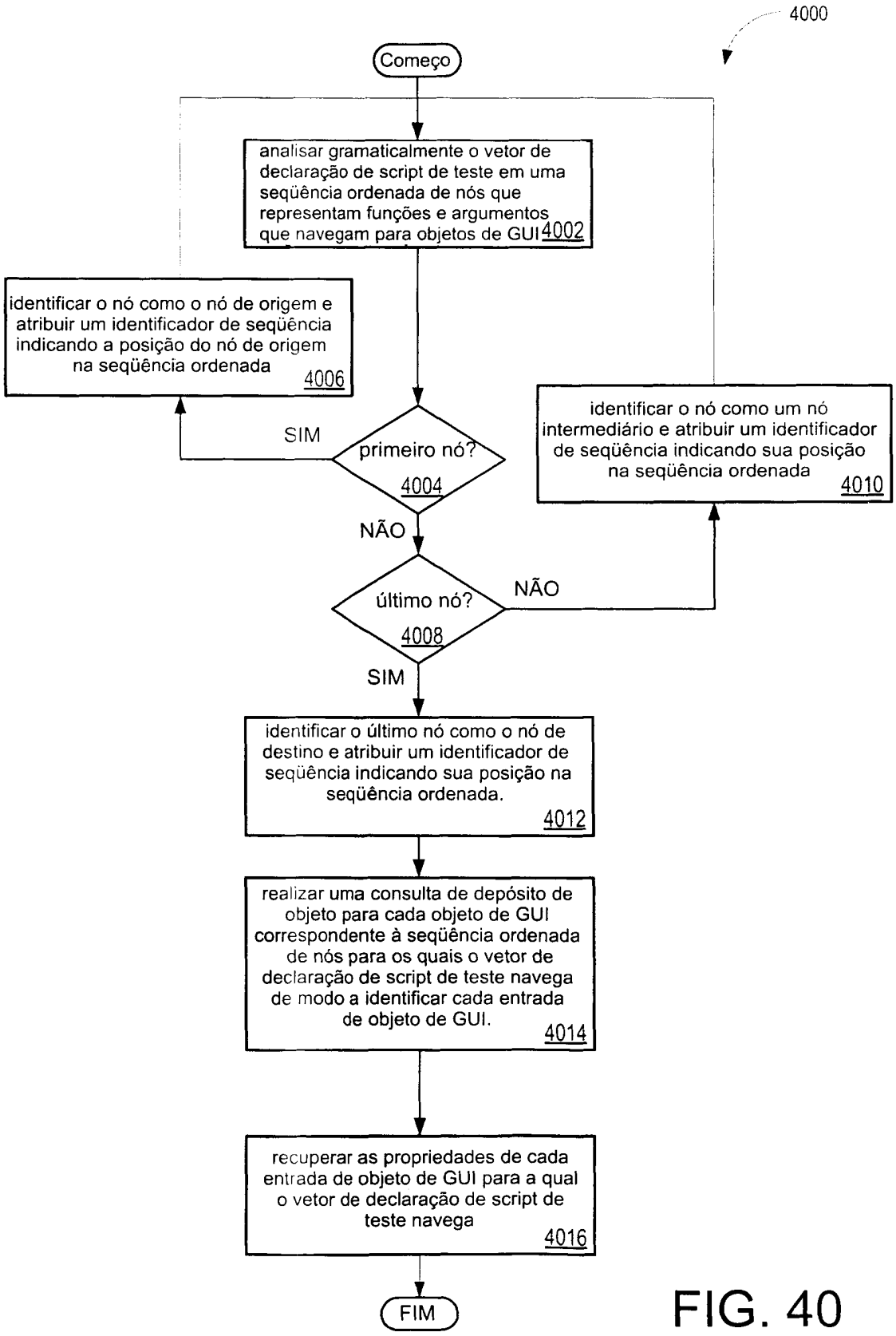

4000
Começo
analisar gramaticalmente o vetor de declaração de script de teste em uma seqüência ordenada de nós que representam funções e argumentos que navegam para objetos de GUI 4002
identificar o nó como o nó de origem e atribuir um identificador de seqüência indicando a posição do nó de origem na seqüência ordenada 4006
SIM
primeiro nó? 4004
identificar o nó como um nó intermediário e atribuir um identificador de seqüência indicando sua posição na seqüência ordenada 4010
NÃO
último nó? 4008
NÃO
SIM
identificar o último nó como o nó de destino e atribuir um identificador de seqüência indicando sua posição na seqüência ordenada. 4012
realizar uma consulta de depósito de objeto para cada objeto de GUI correspondente à seqüência ordenada de nós para os quais o vetor de declaração de script de teste navega de modo a identificar cada entrada de objeto de GUI. 4014
recuperar as propriedades de cada entrada de objeto de GUI para a qual o vetor de declaração de script de teste navega 4016
FIM

FIG. 40

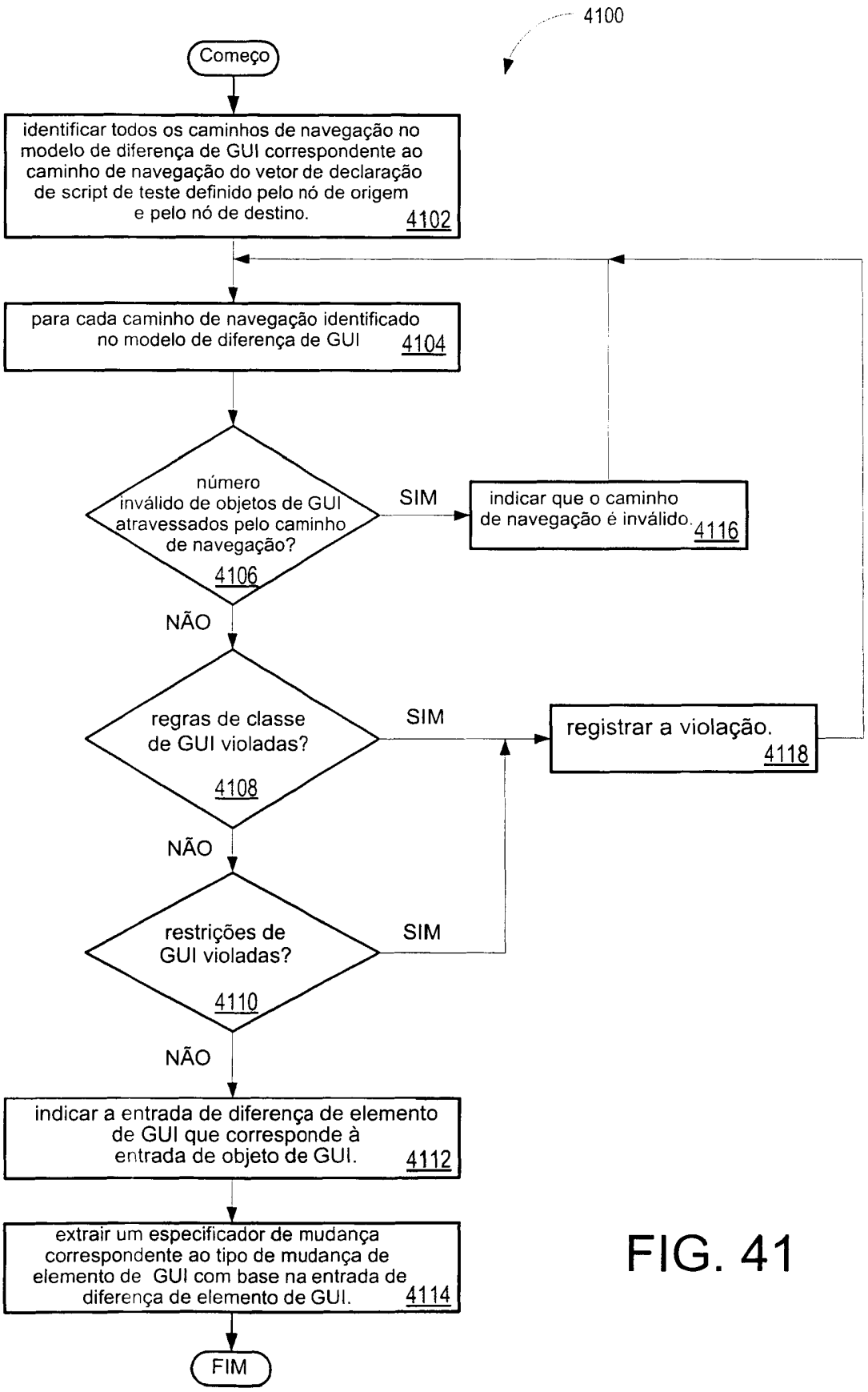

4100
Começo
identificar todos os caminhos de navegação no modelo de diferença de GUI correspondente ao caminho de navegação do vetor de declaração de script de teste definido pelo nó de origem e pelo nó de destino. 4102
para cada caminho de navegação identificado no modelo de diferença de GUI 4104
número inválido de objetos de GUI atravessados pelo caminho de navegação? 4106
SIM
indicar que o caminho de navegação é inválido. 4116
NÃO
regras de classe de GUI violadas? 4108
SIM
registrar a violação. 4118
NÃO
restrições de GUI violadas? 4110
SIM
NÃO
indicar a entrada de diferença de elemento de GUI que corresponde à entrada de objeto de GUI. 4112
extrair um especificador de mudança correspondente ao tipo de mudança de elemento de GUI com base na entrada de diferença de elemento de GUI. 4114
FIM

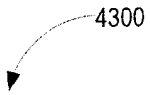
FIG. 41

178

```
'Testar script para o GUI de GAP subseqüente quanto a abrir um arquivo, selecionar uma escola
(Acme State University), mudar a escala acadêmica de 2 para 3 e salvar em um novo arquivo.   178

L1    Window("School").Window("Window").Click 63,15
L2    Window("School").Dialog("Open").WinToolbar("ToolbarWindow32").Press "My Computer"
L3    Window("School").Dialog("Open").WinListView("SysListView32").Activate "Local Disk (C:)"
L4    Window("School").Dialog("Open").WinListView("SysListView32").Activate "University"
L5    Window("School").Dialog("Open").WinComboBox("Files of type:").Select "All files (*.*)"
L6    Window("School").Dialog("Open").WinListView("SysListView32").Select "university.data"
L7    Window("School").Dialog("Open").WinButton("Open").Click
L8    Window("School").WinObject("StateListbox").Click 31,10
L9    Window("School").WinObject("Select State").Click 51,12
L10   Window("School").WinObject("SchoolCombobox").Click 294,14
L11   Window("School").WinEdit("Edit").Set "Acme State University"
L12   Window("School").WinEdit("Edit").Type  micReturn
L13   Window("School").WinObject("2").Drag 15,12
L14   Window("School").WinObject("Academics (1-5)").Drop 73,2
L15   Window("School").WinObject("2").Type "3"
L16   Window("School").WinObject("menuStrip1").Click 101,15
L17   Window("School").Window("Window").Click 69,63
L18   Window("School").WinObject("menuStrip1").Click 97,12
L19   Window("School").Window("Window").Click 52,54
L20   Window("School").Dialog("Save As").WinComboBox("Save as type:").Select "All files (*.*)"
L21   Window("School").Dialog("Save As").WinEdit("File name:").Set "university_revise2.data"
L22   Window("School").Dialog("Save As").WinEdit("File name:").Type  micReturn
```

FIG. 42



Mensagens de guia de mudança:

WP-1: At line 7 of the current test script accesses WinObject ("Select School") that does not exist in the subsequent GAP GUI.

WP-2: At line 6 of the transformed test script may select the wrong GUI object "university.data".

CE-1: At line 4 of the current test script accesses WinObject ("StateListbox") whose default values are changed in the subsequent GAP GUI.

CE-1: At line 8 of the transformed test script accesses WinObject ("StateListbox") whose default values are changed from the current GAP GUI.

CE-2: At line 14 of the transformed test script "Type" a value in a WinObject ("Academics (1-5)") that is read-only.

FIG. 43

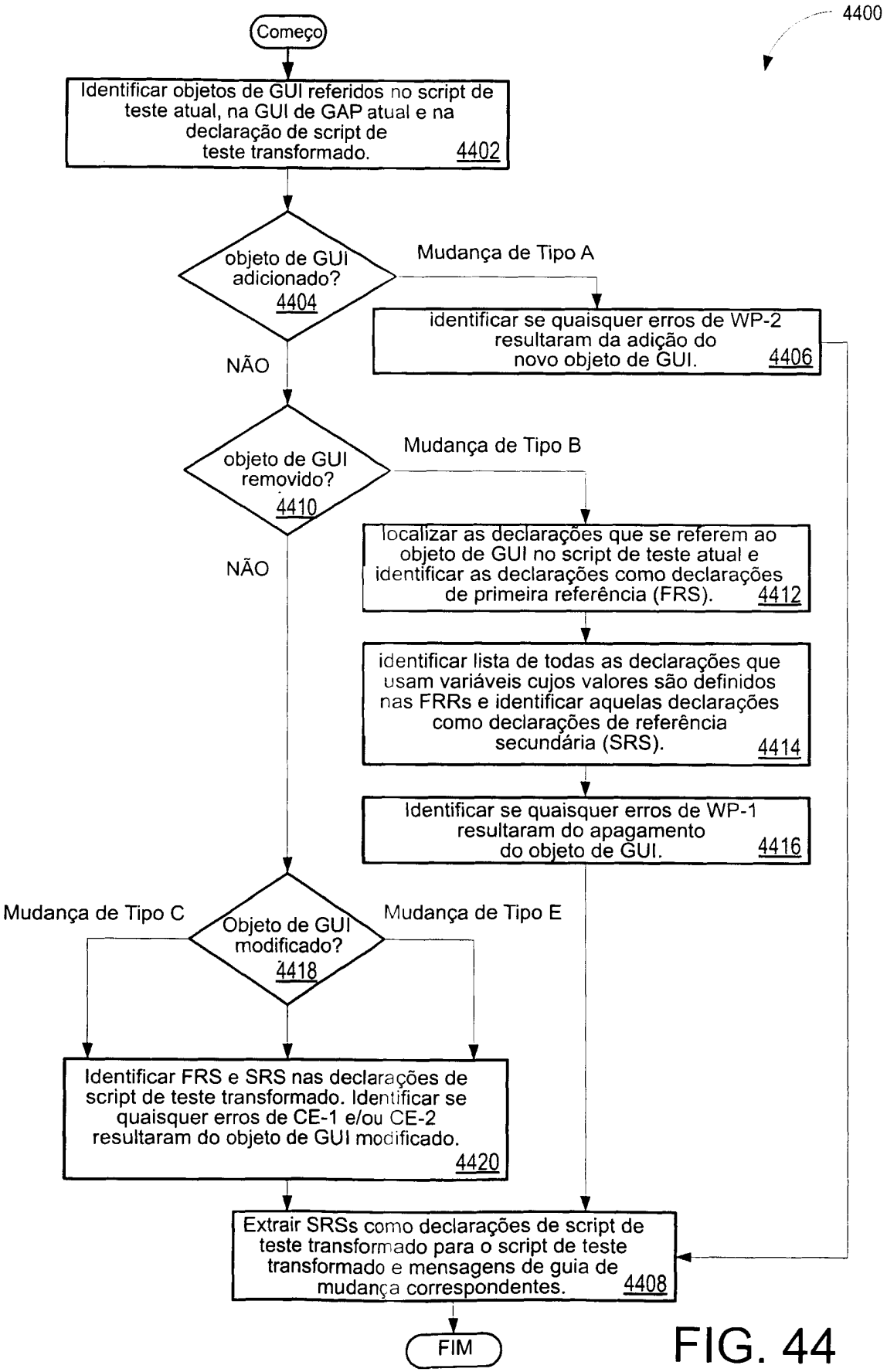

4400
Começo
Identificar objetos de GUI referidos no script de teste atual, na GUI de GAP atual e na declaração de script de teste transformado. 4402
objeto de GUI adicionado? 4404
Mudança de Tipo A
identificar se quaisquer erros de WP-2 resultaram da adição do novo objeto de GUI. 4406
NÃO
objeto de GUI removido? 4410
Mudança de Tipo B
localizar as declarações que se referem ao objeto de GUI no script de teste atual e identificar as declarações como declarações de primeira referência (FRS). 4412
NÃO
identificar lista de todas as declarações que usam variáveis cujos valores são definidos nas FRRs e identificar aquelas declarações como declarações de referência secundária (SRS). 4414
Identificar se quaisquer erros de WP-1 resultaram do apagamento do objeto de GUI. 4416
Mudança de Tipo C
Objeto de GUI modificado? 4418
Mudança de Tipo E
Identificar FRS e SRS nas declarações de script de teste transformado. Identificar se quaisquer erros de CE-1 e/ou CE-2 resultaram do objeto de GUI modificado. 4420
Extrair SRSs como declarações de script de teste transformado para o script de teste transformado e mensagens de guia de mudança correspondentes. 4408
FIM

FIG. 44

4504

```
L1  <GUIElement Type="StateList">

L2   <Version>0
L3    <GUIElement Alias="StateList" Version="0">
        <ParentChildIdentifier>XYZ345</ParentChildIdentifier>
        <UniqueID>0x0</UniqueID>
         <HWND>0x170a64</HWND>
         <Location x="87" y="66" width="792" height="672" />
         <Class>WindowsForms10.Window.8.app4</Class>
L9       <Style>0x16cf0000</Style>
L10      <ExStyle>0xc0050900</ExStyle>
.....
L11   <GUIElement Alias="StateList" Version="0">
     <ParentChildIdentifier>XYZ456</ParentChildIdentifier>
        <UniqueID>0x12</UniqueID>
         <HWND>0x170a64</HWND>
         <Location x="117" y="70" width="784" height="638" />
         <Class>WindowsForms10.Window.8.app4</Class>
          <Style>0x16cf0000</Style>
         <ExStyle>0xc0050900</ExStyle>
.....
           </GUIElement>
          </GUIElement>

   </Version>

L22  <Version>1

L23 <GUIElement Alias="School" Version="1">
        <ParentChildIdentifier>XYZ567</ParentChildIdentifier>
         <UniqueID>0x0</UniqueID>
         <HWND>0x80bd8</HWND>
         <Location x="116" y="88" width="915" height="594" />
         <Class>WindowsForms10.Window.8.app.0.378734a</Class>
L29      <Style>0x16cf0000</Style>
L30      <ExStyle>0xc0050900</ExStyle>
.....
L31  <GUIElement Alias="School" Version="1">
        <ParentChildIdentifier>XYZ789</ParentChildIdentifier>
         <UniqueID>0x12</UniqueID>
         <HWND>0x80bd8</HWND>
         <Location x="146" y="92" width="907" height="560" />
         <Class>WindowsForms10.Window.8.app.0.378734a</Class>
         <Style>0x16cf0000</Style>
         <ExStyle>0xc0050900</ExStyle>
.....
        </GUIElement>
      </GUIElement>
   </Version>
</GUIElement>
```

FIG. 45

4604

```
<GUIElement Type="SchoolListbox">
  <Version>0
L3  - <GUIElement Alias="SchoolListbox" Version="0">
        <ParentChildIdentifier>XYZ321</ParentChildIdentifier>
        <UniqueID>0x3c2</UniqueID>
         <HWND>0x90b52</HWND>
         <Location x="173" y="486" width="336" height="274" />
         <Class>WindowsForms10.LISTBOX.app4</Class>
         <Style>0x560100c1</Style>
         <ExStyle>0xc0000a00</ExStyle>
....
      </GUIElement>
 </Version>

L13   <Version>1
L14 - <GUIElement Alias="SchoolCombobox" Version="1">
          <ParentChildIdentifier>XYZ654</ParentChildIdentifier>
          <UniqueID>0xa8</UniqueID>
          <HWND>0x90e66</HWND>
          <Location x="248" y="653" width="299" height="24" />
          <Class>WindowsForms10.COMBOBOX.app.0.378734a</Class>
          <Style>0x56010242</Style>
           <ExStyle>0xc0000800</ExStyle>
  ..........
       </GUIElement>

 </Version>
```

FIG. 46

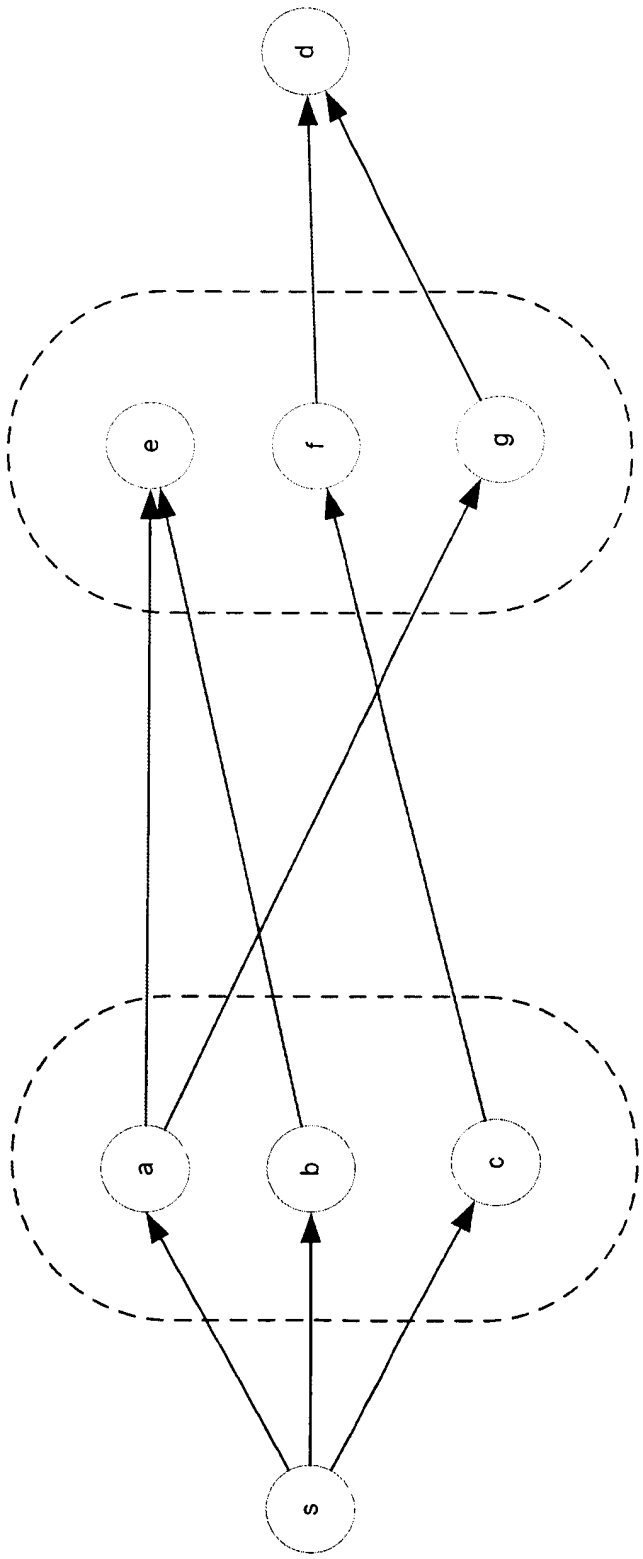


FIG. 47

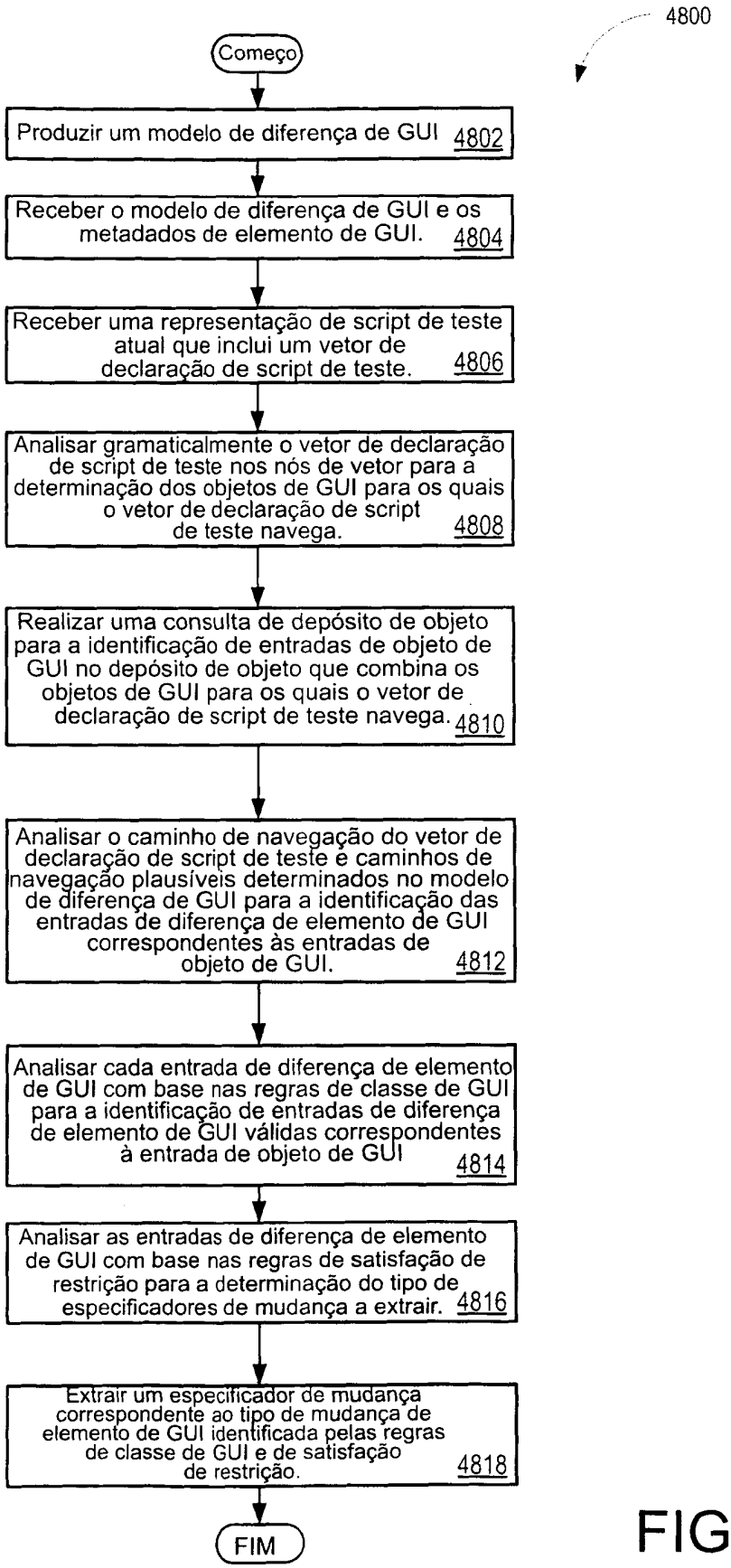

4800
Começo
Produzir um modelo de diferença de GUI 4802
Receber o modelo de diferença de GUI e os metadados de elemento de GUI. 4804
Receber uma representação de script de teste atual que inclui um vetor de declaração de script de teste. 4806
Analisar gramaticalmente o vetor de declaração de script de teste nos nós de vetor para a determinação dos objetos de GUI para os quais o vetor de declaração de script de teste navega. 4808
Realizar uma consulta de depósito de objeto para a identificação de entradas de objeto de GUI no depósito de objeto que combina os objetos de GUI para os quais o vetor de declaração de script de teste navega. 4810
Analisar o caminho de navegação do vetor de declaração de script de teste e caminhos de navegação plausíveis determinados no modelo de diferença de GUI para a identificação das entradas de diferença de elemento de GUI correspondentes às entradas de objeto de GUI. 4812
Analisar cada entrada de diferença de elemento de GUI com base nas regras de classe de GUI para a identificação de entradas de diferença de elemento de GUI válidas correspondentes à entrada de objeto de GUI 4814
Analisar as entradas de diferença de elemento de GUI com base nas regras de satisfação de restrição para a determinação do tipo de especificadores de mudança a extrair. 4816
Extrair um especificador de mudança correspondente ao tipo de mudança de elemento de GUI identificada pelas regras de classe de GUI e de satisfação de restrição. 4818
FIM

FIG. 48

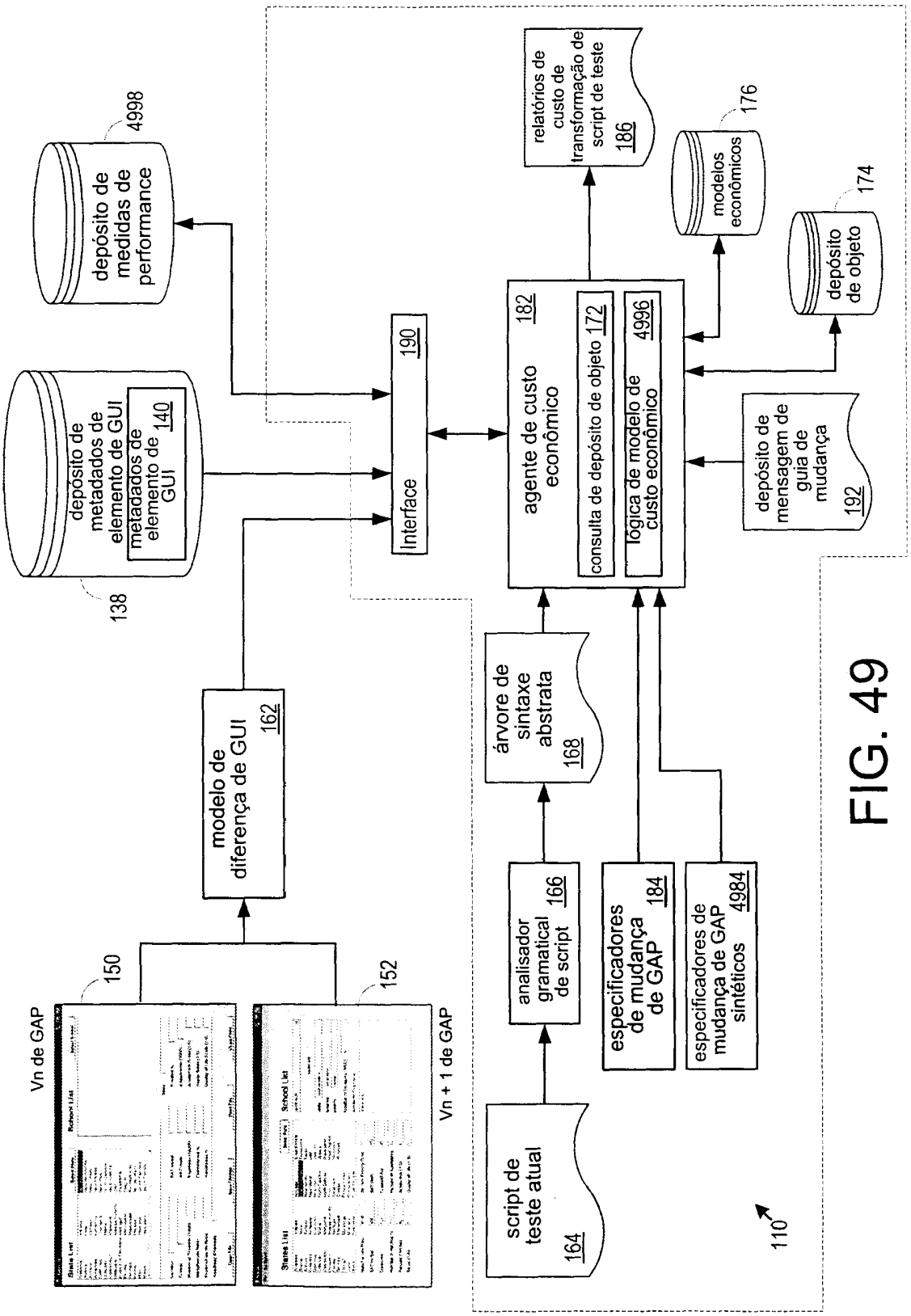

Vn de GAP
150
Vn + 1 de GAP
152
modelo de diferença de GUI 162
depósito de metadados de elemento de GUI
metadados de elemento de GUI 140
138
depósito de medidas de performance
4998
Interface 190
agente de custo econômico 182
consulta de depósito de objeto 172
lógica de modelo de custo econômico 4996
relatórios de custo de transformação de script de teste 186
modelos econômicos 176
depósito de objeto 174
depósito de mensagem de guia de mudança 192
árvore de sintaxe abstrata 168
analisador gramatical de script 166
especificadores de mudança de GAP 184
especificadores de mudança de GAP sintéticos 4984
script de teste atual 164
110

FIG. 49

```
' Testar script quanto a GUI de GAP atual para abertura de arquivo, seleção de várias escolas
' (por exemplo, Acme University e State University), mudar a escala acadêmica para 3 para a      5000
' Acme University e para 2 para a State University no mesmo estado identificado pela coordenada
  36, 14 na linha 5 e salvar em um novo arquivo.

L1   Window("StateList").WinObject("Open File").Click 86,12
L2   Window("StateList").Dialog("Open").WinListView("SysListView32").Select "university.data"
L3   Window("StateList").Dialog("Open").WinButton("Open").Click
L4   Window("StateList").WinObject("StateListbox").Click 31,7
L5   Window("StateList").WinObject("Select State").Click 36,14
L6   Window("StateList").WinObject("SchoolListbox").Click 19,22

L7   For School_Constant = J to K

L8      Window("StateList").WinObject("Select School").Click 67, School_Constant
L9      Window("StateList").WinObject("AcadScale").Drag 18,14
L10     Window("StateList").Drop 698,471

L11     If Window("StateList").WinObject("Select School").property("value") = "Acme University" then
L12        Window("StateList").WinObject("AcadScale").Type "3"
L13     Else if Window("StateList").WinObject("Select School").property("value") = "State University" then
L14        Window("StateList").WinObject("AcadScale").Type "2"
L15     End if

L16     Window("StateList").WinObject("Save Change").Click 108,7

L17  Next School_Constant

L18  Window("StateList").WinObject("Save File").Click 70,16
L19  Window("StateList").Dialog("Save a Data Record").WinEdit("File name:").Set "university_revise.data"
L20  Window("StateList").Dialog("Save a Data Record").WinButton("Save").Click
```

5000

FIG. 50

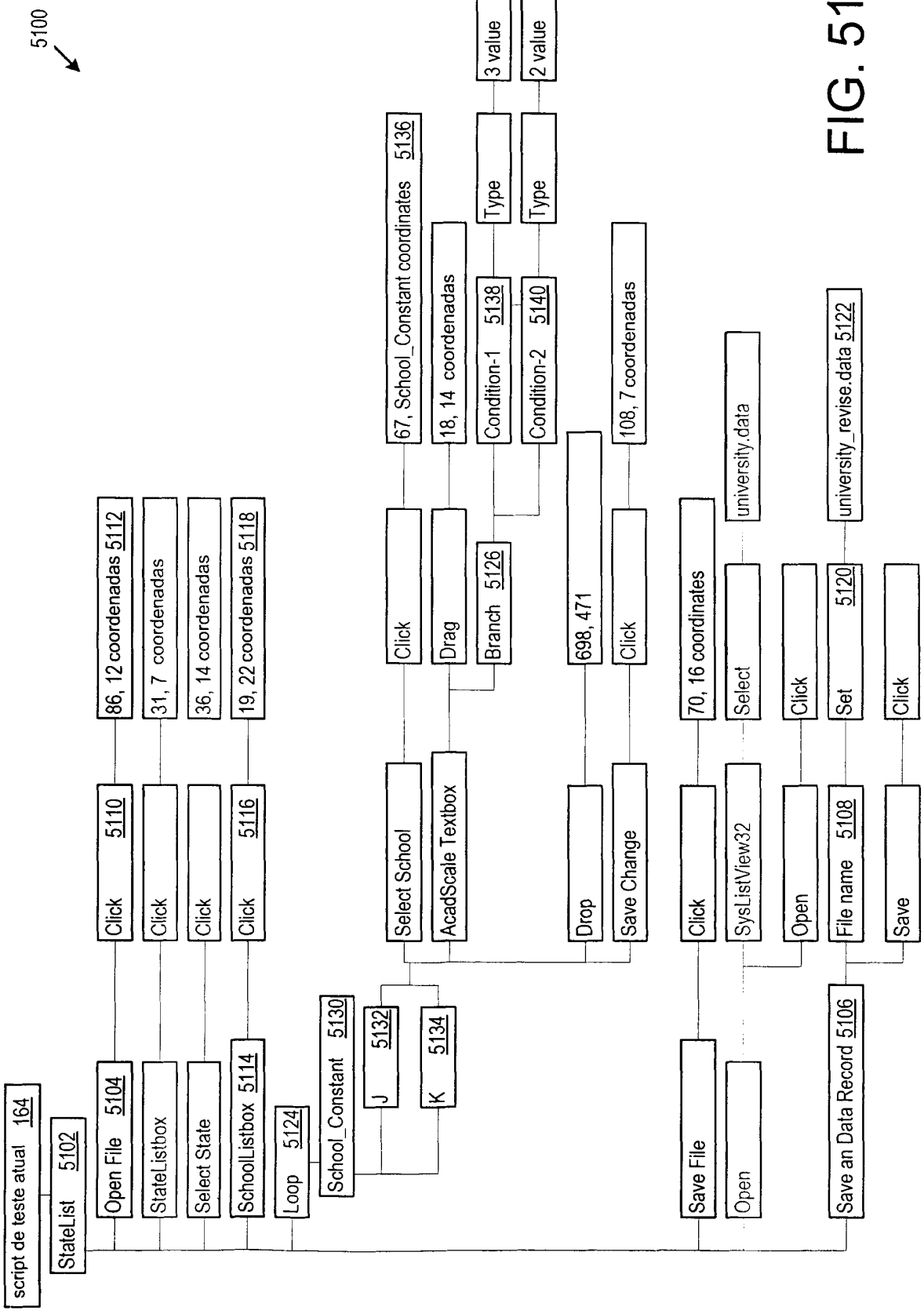

5100
script de teste atual 164
StateList 5102
Open File 5104
StateListbox
Select State
SchoolListbox 5114
Loop 5124
School_Constant 5130
J 5132
K 5134
Click 5110
Click
Click
Click 5116
86, 12 coordenadas 5112
31, 7 coordenadas
36, 14 coordenadas
19, 22 coordenadas 5118
Select School
AcadScale Textbox
Click
Drag
67, School_Constant coordinates 5136
18, 14 coordenadas
Branch 5126
Condition-1 5138
Condition-2 5140
Type
Type
3 value
2 value
Drop
Save Change
698, 471
108, 7 coordenadas
Save File
Open
Click
SysListView32
Open
Click
70, 16 coordinates
Select
university.data
Save an Data Record 5106
File name 5108
Set 5120
Save
Click
university_revise.data 5122

FIG. 51

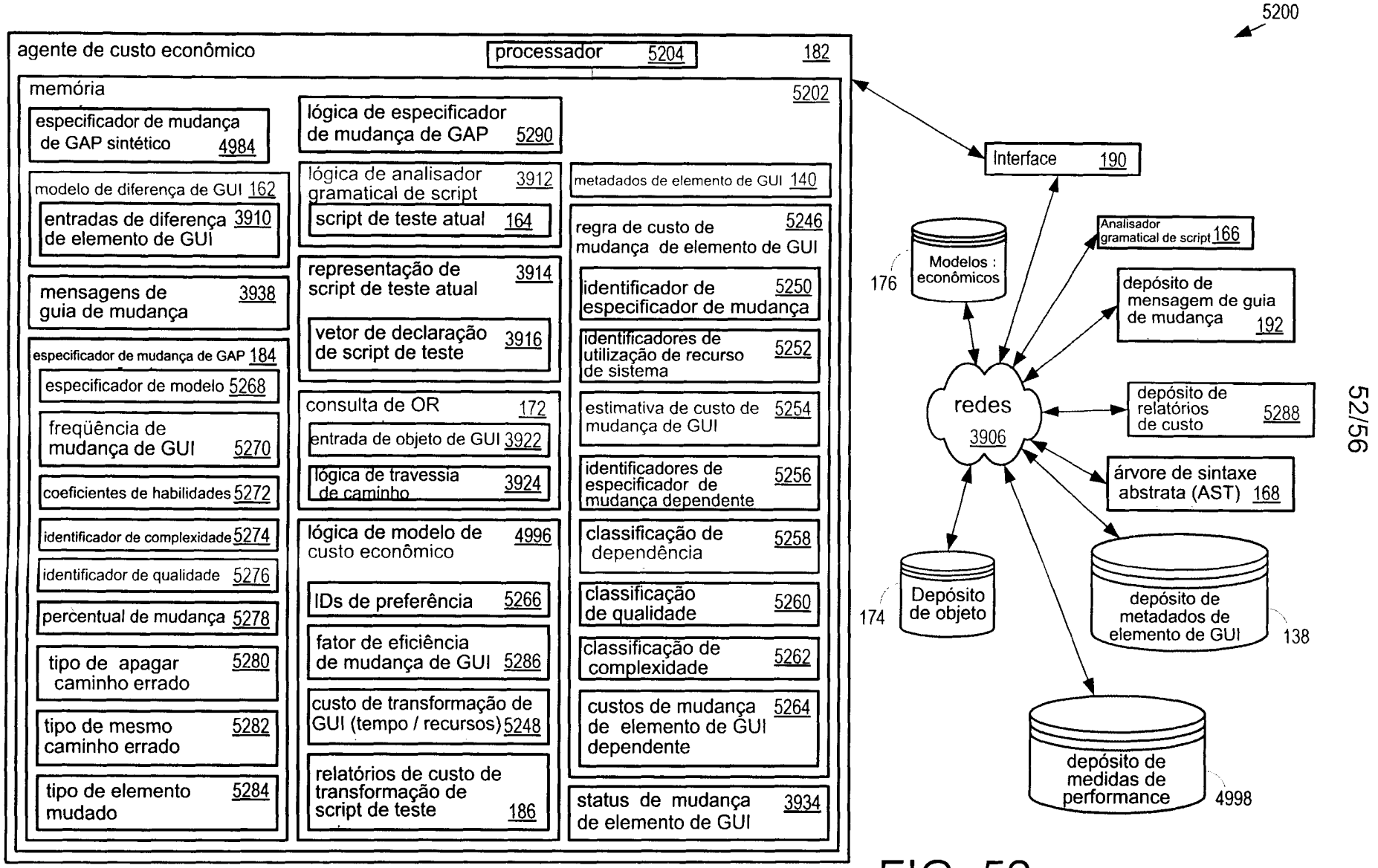

5200
agente de custo econômico
processador 5204
182
memória
5202
especificador de mudança de GAP sintético 4984
modelo de diferença de GUI 162
entradas de diferença de elemento de GUI 3910
mensagens de guia de mudança 3938
especificador de mudança de GAP 184
especificador de modelo 5268
freqüência de mudança de GUI 5270
coeficientes de habilidades 5272
identificador de complexidade 5274
identificador de qualidade 5276
percentual de mudança 5278
tipo de apagar caminho errado 5280
tipo de mesmo caminho errado 5282
tipo de elemento mudado 5284
lógica de especificador de mudança de GAP 5290
lógica de analisador gramatical de script 3912
script de teste atual 164
representação de script de teste atual 3914
vetor de declaração de script de teste 3916
consulta de OR 172
entrada de objeto de GUI 3922
lógica de travessia de caminho 3924
lógica de modelo de custo econômico 4996
IDs de preferência 5266
fator de eficiência de mudança de GUI 5286
custo de transformação de GUI (tempo / recursos) 5248
relatórios de custo de transformação de script de teste 186
metadados de elemento de GUI 140
regra de custo de mudança de elemento de GUI 5246
identificador de especificador de mudança 5250
identificadores de utilização de recurso de sistema 5252
estimativa de custo de mudança de GUI 5254
identificadores de especificador de mudança dependente 5256
classificação de dependência 5258
classificação de qualidade 5260
classificação de complexidade 5262
custos de mudança de elemento de GUI dependente 5264
status de mudança de elemento de GUI 3934
Interface 190
Modelos : econômicos
176
Analisador gramatical de script 166
depósito de mensagem de guia de mudança 192
redes 3906
depósito de relatórios de custo 5288
árvore de sintaxe abstrata (AST) 168
Depósito de objeto
174
depósito de metadados de elemento de GUI
138
depósito de medidas de performance
4998

FIG. 52

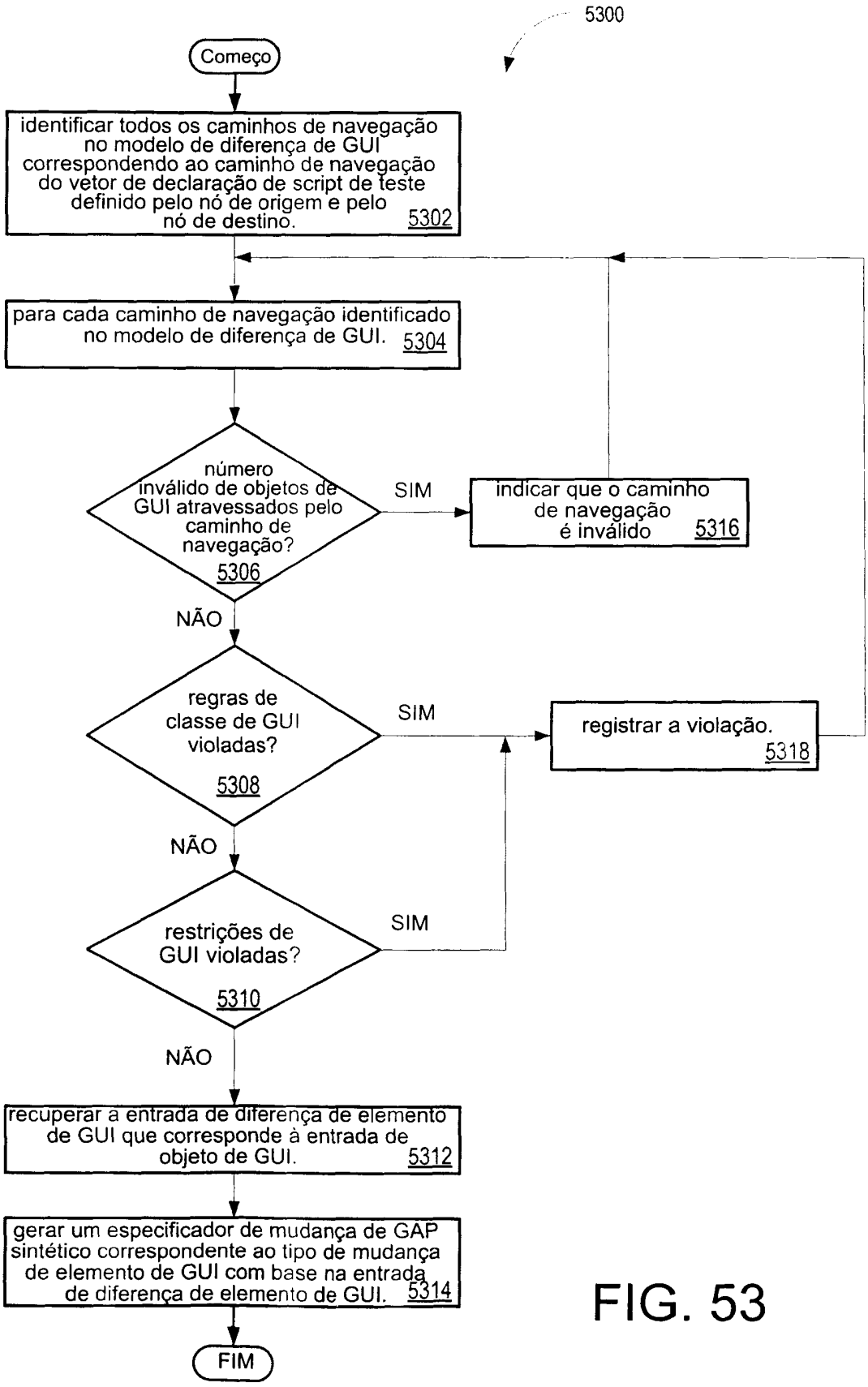

5300
Começo
identificar todos os caminhos de navegação no modelo de diferença de GUI correspondendo ao caminho de navegação do vetor de declaração de script de teste definido pelo nó de origem e pelo nó de destino. 5302
para cada caminho de navegação identificado no modelo de diferença de GUI. 5304
número inválido de objetos de GUI atravessados pelo caminho de navegação? 5306
SIM
indicar que o caminho de navegação é inválido 5316
NÃO
regras de classe de GUI violadas? 5308
SIM
registrar a violação. 5318
NÃO
restrições de GUI violadas? 5310
SIM
NÃO
recuperar a entrada de diferença de elemento de GUI que corresponde à entrada de objeto de GUI. 5312
gerar um especificador de mudança de GAP sintético correspondente ao tipo de mudança de elemento de GUI com base na entrada de diferença de elemento de GUI. 5314
FIM

FIG. 53

5400

```
' Testar script quanto a GUI de GAP subseqüente para abertura de arquivo, seleção
' de várias escolas (por exemplo, Acme University e State University), mudar a escala
' acadêmica para 3 para a Acme University e para 2 para a State University e salvar
  em um novo arquivo.                                                          5400
L1    Window("School").Window("Window").Click 63,15
L2    Window("School").Dialog("Open").WinToolbar("ToolbarWindow32").Press "My Computer"
L3    Window("School").Dialog("Open").WinListView("SysListView32").Activate "Local Disk (C:)"
L4    Window("School").Dialog("Open").WinListView("SysListView32").Activate "University"
L5    Window("School").Dialog("Open").WinComboBox("Files of type:").Select "All files (*.*)"
L6    Window("School").Dialog("Open").WinListView("SysListView32").Select "university.data"
L7    Window("School").Dialog("Open").WinButton("Open").Click
L8    Window("School").WinObject("StateListbox").Click 31,10
L9    Window("School").WinObject("Select State").Click 51,12
L10   Window("School").WinObject("SchoolCombobox").Click 294,14

L11   For School_Constant = J to K
L12       If School_Constant = J then
L13           School_Name == "Acme University"
L14           Acad_Scale == "3"
L15       Elseif School_Constant = K then
L16           School_Name == "State University"
L17           Acad_Scale == "2"
L18       End if
L19       Window("School").WinEdit("Edit").Set School_Name
L20       Window("School").WinEdit("Edit").Type  micReturn
L21       Window("School").WinObject("2").Drag 15,12
L22       Window("School").WinObject("Academics (1-5)").Drop 73,2
L23       Window("School").WinObject("2").Type Acad_Scale
L24       Window("School").WinObject("menuStrip1").Click 101,15
L25       Window("School").Window("Window").Click 69,63
L26   Next School_Constant

L27   Window("School").WinObject("menuStrip1").Click 97,12
L28   Window("School").Window("Window").Click 52,54
L29   Window("School").Dialog("Save As").WinComboBox("Save as type:").Select "All files (*.*)"
L30   Window("School").Dialog("Save As").WinEdit("File name:").Set "university_revise2.data"
L31   Window("School").Dialog("Save As").WinEdit("File name:").Type  micReturn
```

FIG. 54

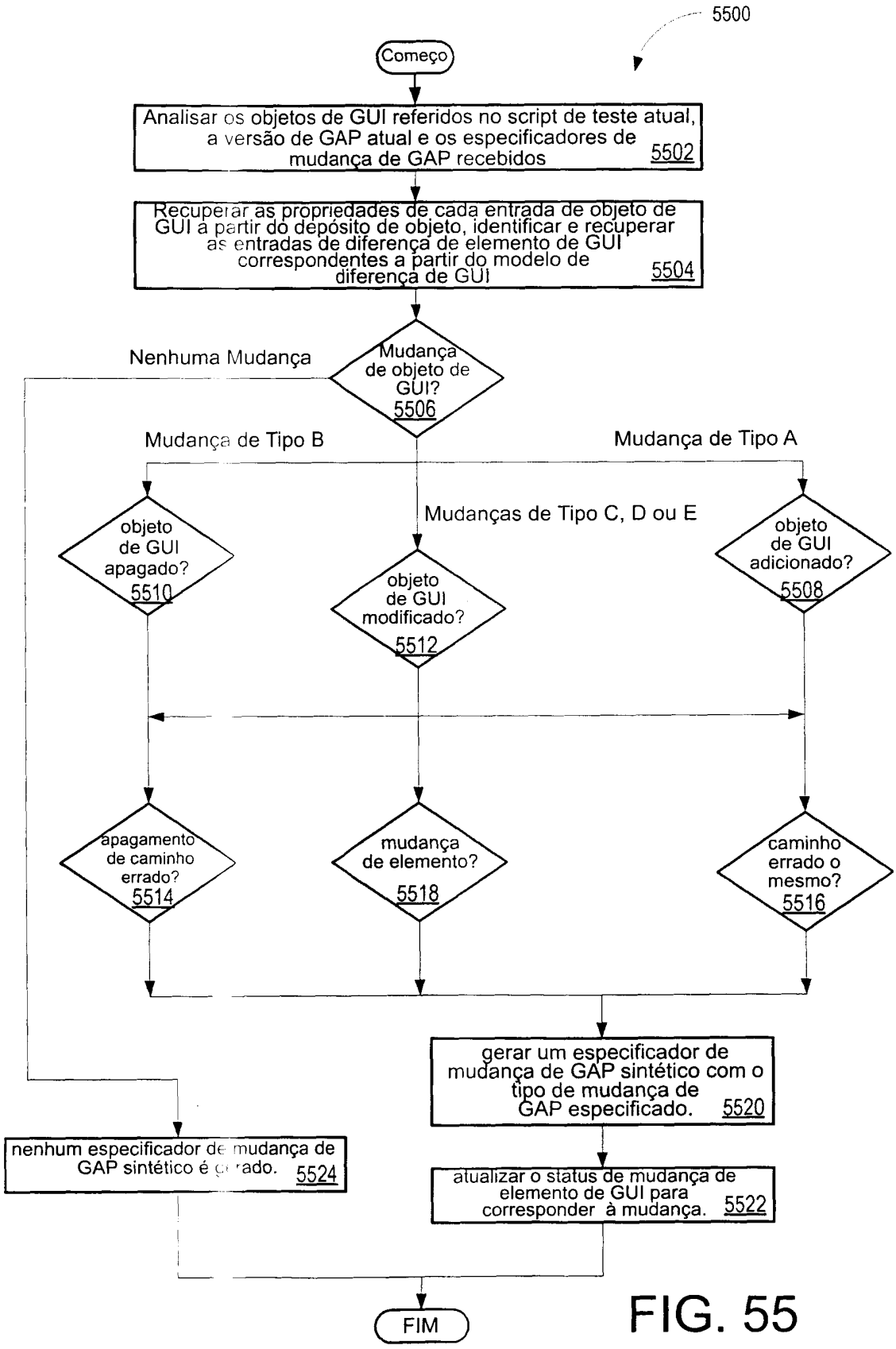

5500
Começo
Analisar os objetos de GUI referidos no script de teste atual, a versão de GAP atual e os especificadores de mudança de GAP recebidos 5502
Recuperar as propriedades de cada entrada de objeto de GUI a partir do depósito de objeto, identificar e recuperar as entradas de diferença de elemento de GUI correspondentes a partir do modelo de diferença de GUI 5504
Mudança de objeto de GUI? 5506
Nenhuma Mudança
Mudança de Tipo B
Mudanças de Tipo C, D ou E
Mudança de Tipo A
objeto de GUI apagado? 5510
objeto de GUI modificado? 5512
objeto de GUI adicionado? 5508
apagamento de caminho errado? 5514
mudança de elemento? 5518
caminho errado o mesmo? 5516
gerar um especificador de mudança de GAP sintético com o tipo de mudança de GAP especificado. 5520
nenhum especificador de mudança de GAP sintético é gerado. 5524
atualizar o status de mudança de elemento de GUI para corresponder à mudança. 5522
FIM

FIG. 55

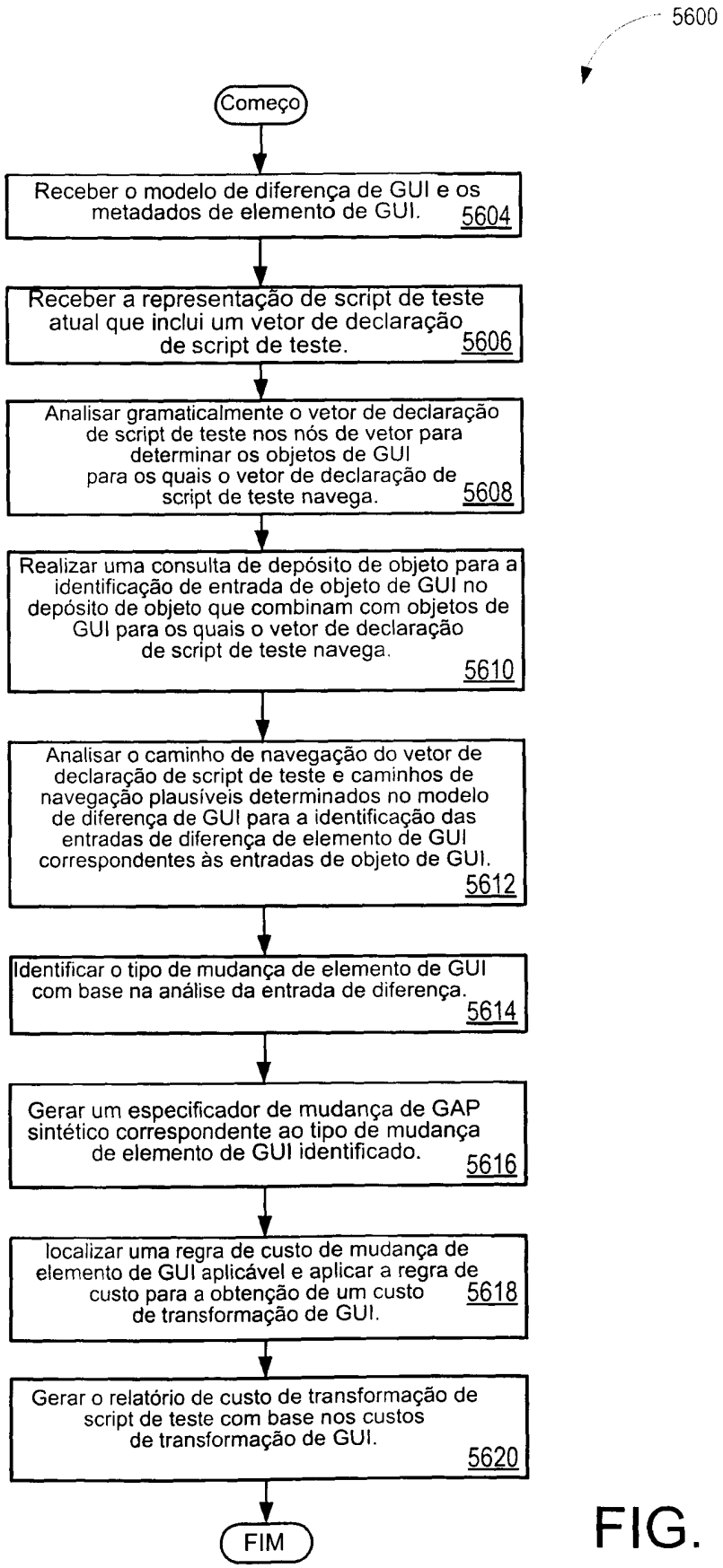

5600
Começo
Receber o modelo de diferença de GUI e os metadados de elemento de GUI. 5604
Receber a representação de script de teste atual que inclui um vetor de declaração de script de teste. 5606
Analisar gramaticalmente o vetor de declaração de script de teste nos nós de vetor para determinar os objetos de GUI para os quais o vetor de declaração de script de teste navega. 5608
Realizar uma consulta de depósito de objeto para a identificação de entrada de objeto de GUI no depósito de objeto que combinam com objetos de GUI para os quais o vetor de declaração de script de teste navega. 5610
Analisar o caminho de navegação do vetor de declaração de script de teste e caminhos de navegação plausíveis determinados no modelo de diferença de GUI para a identificação das entradas de diferença de elemento de GUI correspondentes às entradas de objeto de GUI. 5612
Identificar o tipo de mudança de elemento de GUI com base na análise da entrada de diferença. 5614
Gerar um especificador de mudança de GAP sintético correspondente ao tipo de mudança de elemento de GUI identificado. 5616
localizar uma regra de custo de mudança de elemento de GUI aplicável e aplicar a regra de custo para a obtenção de um custo de transformação de GUI. 5618
Gerar o relatório de custo de transformação de script de teste com base nos custos de transformação de GUI. 5620
FIM

FIG. 56