(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 913 509**

(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **07 01622**

(51) Int Cl[8] : ***G 06 F 9/44*** *(2006.01), G 06 F 19/00, G 06 Q 90/00*

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 06.03.07.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 12.09.08 Bulletin 08/37.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *THALES Société anonyme* — FR.

(72) **Inventeur(s) :** FAURE DAVID, LARD JEROME, GRISVARD OLIVIER et SEDOGBO CELESTIN.

(73) **Titulaire(s) :**

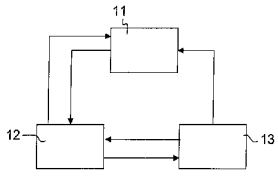
(74) **Mandataire(s) :** MARKS & CLERK FRANCE.

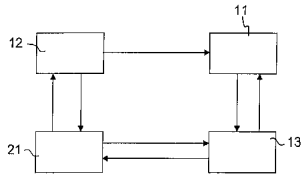
(54) PROCEDE DE GESTION D'INTERACTIONS ENTRE DES APPLICATIONS ET DES UTILISATEURS.

(57) L'invention concerne un procédé de gestion de l'interaction d'une application avec au moins un utilisateur, le comportement de ladite application étant dépendant du contexte d'utilisation, défini notamment par un historique de l'activité de l'utilisateur, comportant:

- une étape de création d'au moins un modèle, ledit modèle représentant de façon dynamique l'application et ses données,
- une étape de création d'au moins une vue, ladite vue présentant les données du modèle à l'utilisateur et capturant les évènements utilisateurs,
- une étape de création d'au moins un contrôleur, ledit contrôleur définissant le comportement de l'application en mettant en correspondance les évènements utilisateurs avec des altérations du modèle,

caractérisé en ce qu'il comporte en outre une étape de création d'un module d'interaction effectuant l'interprétation sémantique des évènements échangés entre la vue et le contrôleur, ladite interprétation sémantique permettant d'adapter de vérifier la cohérence entre les interactions de l'utilisateur et le contexte d'utilisation.





FR 2 913 509 - A1

Procédé de gestion d'interactions entre des applications et des utilisateurs

L'invention concerne un procédé de gestion d'interactions entre une application et un ou plusieurs utilisateurs.

De nombreux systèmes informatiques sont actuellement basés sur le patron de conception MVC (Modèle, Vue, Contrôleur). Un patron est une solution générale à un problème de conception récurrent. Ce patron de conception permet un développement modulaire d'une application interactive en trois parties plus ou moins indépendantes selon les techniques de développement. Il permet de concevoir un logiciel interactif robuste aux évolutions comprenant : une partie concernant les données de l'application (le modèle), une partie comportement de l'application (le contrôleur) et une dernière partie pour la visualisation du comportement et des données de l'application à la fois (la vue).

Le patron de conception MVC définit la répartition des fonctionnalités nécessaires à une application au sein de ces trois éléments. Le schéma de ce patron de conception est illustré en Figure 1. Chacune des parties a une vocation spécifique.

Le modèle 11 a pour objectif de proposer une représentation des fonctions d'une application et de ses données. Il est en mesure de répondre aux requêtes exprimées sur ces éléments et d'informer la vue 12 sur les modifications des éléments auxquels elle est abonnée.

La vue 12 a pour objectif de présenter le modèle 11 et de transmettre les évènements utilisateur au contrôleur 13. Elle peut demander des informations au modèle 11 et doit autoriser le contrôleur 13 à la sélectionner.

Le contrôleur 13 définit le comportement de l'application. Il a pour charge de mettre en correspondance les évènements utilisateur avec des altérations du modèle 11. Il doit de plus sélectionner la vue 12 pour la présentation des données du modèle.

Le patron de conception MVC a été développé dans un contexte de conception centré sur le développement d'applications mono-utilisateur, mono-surface (un poste de travail unique par utilisateur) et à domaine de

compétence unique (i.e. le développement d'applications pour le travail de secrétariat).

De ce fait, l'utilisation de ce patron ne permet pas de prendre en compte facilement différentes formes d'interactions avec les utilisateurs d'une application ou les préférences spécifiques d'un utilisateur. De plus, ce patron ne permet pas non plus de représenter de manière riche les spécificités du contexte d'interaction (domaine d'application, spécificités de l'utilisateur, de l'organisation, ...). En effet, de nombreux domaines d'applications présentent à la fois des contraintes importantes sur la présentation de l'information (visualisation d'information) ainsi que sur les modèles du domaine eux-mêmes. Cette approche de conception « centrée utilisateur » et « centrée domaine/métier » requiert la modification du patron de conception MVC. L'invention impacte toutes les catégories de systèmes où l'humain tient une place prépondérante et ceci dans toutes sortes de domaines d'application, tels que :

- La défense : systèmes d'armes, de commandement, de simulation et d'entraînement ;
- Les transports : systèmes de pilotage, de supervision, de réservation ;
- Les communications : téléphonie, radio, télévision, internet ;
- La vie quotidienne : systèmes électroménagers, véhicules individuels, informatique domestique omniprésente et diffuse (« ubiquitous computing » en anglais) ;
- Les services : systèmes bancaires, commerce électronique, assistance technique ;
- La santé : systèmes hospitaliers, secours opérationnel.

Plusieurs problèmes se posent avec le patron de conception MVC utilisé dans de tels systèmes. La gestion de l'interaction entre l'utilisateur et l'application est diffuse. Elle est répartie et gérée au sein de plusieurs modules. Par exemple, le contrôleur décide de sélectionner une vue pour présenter une information du modèle mais la vue est abonnée aux informations qui l'intéressent dans le modèle. Le contrôleur n'a ainsi pas la possibilité de gérer les informations qui seront présentées.

Le changement du comportement de l'application géré par le contrôleur impose tout d'abord un changement de contrôleur. Ainsi, une modification du rôle d'un opérateur imposerait le changement du contrôleur déployé dans le système qu'il utilise. L'application n'a en effet pas le même comportement.

Le développement d'une application centrée utilisateur est complexe à réaliser avec ce patron de conception dans la mesure où il est nécessaire que l'ensemble du système soit spécifique à l'utilisateur. Cette spécificité du système implique une vue, un contrôleur et un modèle spécifique à chaque utilisateur.

L'invention vise à pallier les problèmes cités précédemment en proposant un procédé permettant d'ajouter un module d'interaction entre la vue et le contrôleur. Ce module d'interaction a en charge la gestion des informations spécifiques à un utilisateur particulier afin de permettre l'indépendance du modèle, de la vue et du contrôleur par rapport à l'utilisateur, mais également à la surface d'interaction et enfin permet une évolution simplifiée d'un domaine d'application à un autre. Cette invention permet l'utilisation d'une approche basée sur des modèles externes aux modules d'interaction et de contrôle permettant d'accroître leur réutilisabilité.

Le principal avantage de cette invention est de permettre une interaction centrée utilisateur et centrée domaine. La gestion de l'interaction avec l'utilisateur est réunie dans le module d'interaction. Cette approche centralise les aspects relatifs à l'utilisateur dans le module de l'interaction afin de permettre au contrôleur de se focaliser sur le contrôle du comportement applicatif au contraire de celui de l'utilisateur.

Le deuxième intérêt de cette invention est de permettre l'interaction basée sur des modèles externes. La gestion de l'interaction est basée sur des modèles de représentation externes à l'application telles que les tâches recommandées pour l'utilisateur ou les préférences dans les modes d'utilisation. Ces modèles externes peuvent être des modèles de tâches ou des profils pour l'interaction. Le module interaction utilisera ces modèles externes par l'intermédiaire d'un gestionnaire de dialogue lui-même basé sur un modèle de dialogue. Cette approche basée sur des modèles augmente l'aspect générique des différents modules contrôleur, interaction et

vue. En effet, les différents modules mettent en œuvre des processus utilisant les descriptions des interactions exprimées dans les modèles lors de l'exécution de l'application.

Le troisième intérêt est de proposer une généralisation accrue des modules vue et contrôleur. Une plus forte généralisation signifie donc une simplification des rôles attribués aux modules vue et contrôleur. Leurs responsabilités dans le développement d'une application informatique interactive sont donc plus claires. La gestion de l'interaction homme-machine, auparavant réalisée dans la vue ou dans le contrôle est déléguée entièrement au module d'interaction. Par exemple, la requête de la vue vers une donnée du modèle passant par le contrôleur permettra d'utiliser la même vue quel que soit le modèle utilisé puisque le contrôleur se chargera de cette adaptation grâce au module d'interaction.

Le quatrième intérêt de cette invention est l'introduction du module d'interaction augmentant l'aspect générique du contrôleur. Le contrôleur peut être réutilisé quelle que soit l'interaction proposée à l'utilisateur. Seul les modèles du module d'interaction seront altérés. Par exemple, le changement de rôle d'un utilisateur ne nécessitera que le changement des modèles de l'interaction.

Pour résumer, cette invention permet l'augmentation de la modularité d'un modèle classique MVC en introduisant l'approche centrée utilisateur et la gestion de l'interaction et des adaptations basées sur des modèles externes à l'application. Cette approche permet à la vue, au modèle d'être réutilisables quel que soit l'utilisateur ainsi que le domaine visé. Enfin, les modules contrôleur et Interaction sont réutilisables et configurables dans la mesure où ils sont basés sur des modèles.

À cet effet, l'invention a pour objet un procédé de gestion de l'interaction d'au moins une application avec au moins un utilisateur, le comportement de ladite application étant dépendant du contexte d'utilisation, défini notamment par un historique de l'activité de l'utilisateur, comportant :

- une étape de création d'au moins un modèle, ledit modèle représentant de façon dynamique l'application et ses données,

- une étape de création d'au moins une vue, ladite vue présentant les données du modèle à l'utilisateur et capturant les évènements utilisateurs,
- une étape de création d'au moins un contrôleur, ledit contrôleur définissant le comportement de l'application en mettant en correspondance les évènements utilisateurs avec des altérations du modèle,

caractérisé en ce qu'il comporte en outre une étape de création d'un module d'interaction effectuant l'interprétation sémantique des évènements échangés entre la vue et le contrôleur, ladite interprétation sémantique permettant d'adapter les évènements échangés et de vérifier la cohérence entre les interactions de l'utilisateur et le contexte d'utilisation.

Avantageusement, le module d'interaction comprend notamment :

- des moyens pour transmettre des évènements entre la vue le contrôleur,
- des moyens pour créer de nouveaux évènements,
- des moyens pour filtrer, supprimer et altérer des évènements,
- des moyens pour adapter les informations en provenance de l'utilisateur ou vers l'utilisateur.

Avantageusement, le module d'interaction comprend en outre des moyens pour transmettre des évènements entre la vue et le contrôleur en introduisant un décalage temporel.

Avantageusement, le module d'interaction est réalisé avec un modèle statique, défini à la conception de l'application.

Avantageusement, le module d'interaction est réalisé avec un modèle dynamique pouvant être modifié lors de l'exécution de l'application. La prise en compte des modifications dans le modèle peut être immédiate ou non.

Avantageusement, le module d'interaction réalise l'interprétation sémantique des événements à l'aide d'un modèle de tâche, ledit modèle de tâche comprenant notamment :

- un enchaînement de tâches à réaliser par l'utilisateur,
- des événements pouvant survenir,
- une description des interactions à effectuer lorsqu'un événement survient,
- les conditions de déclenchement nécessaires desdites interactions,
- les informations à fournir en retour par lesdites interactions,
- un état courant de l'utilisateur dans sa tâche.

Avantageusement, le module d'interaction comprend : un gestionnaire de tâche pour gérer en temps réel les modèles de tâches, une base de donnée pour stocker les modèles de tâches.

Avantageusement, le module d'interaction comprend en outre un gestionnaire de dialogue faisant l'interface entre le module d'interaction, la vue et le contrôleur.

L'invention sera mieux comprise et d'autres avantages apparaîtront à la lecture de la description détaillée et à l'aide des figures parmi lesquelles :

La figure 1 représente un patron de conception MVC selon l'art connu.

La figure 2 illustre un patron de conception obtenu avec le procédé selon l'invention comprenant un modèle, une vue, un contrôleur et un module interaction.

La figure 3 représente un exemple de modèle de tâches.

La figure 4 montre un exemple de connexion d'un utilisateur à un système réalisé en suivant le procédé selon l'invention.

L'invention concerne un procédé permettant d'ajouter un module d'interaction entre une vue et un contrôleur. Le patron ainsi obtenu est illustré par la figure 2. Le module d'interaction 21 effectue l'interprétation sémantique

des évènements échangés entre la vue 12 et le contrôleur 13. Pour ce faire, le module d'interaction 21 mémorise un contexte d'interaction. L'interprétation sémantique permet de vérifier ou d'adapter la cohérence entre les interactions de l'utilisateur et le contexte d'utilisation. Le module d'interaction comprend notamment :

- des moyens pour transmettre des évènements entre la vue 12 le contrôleur 13 en insérant éventuellement un décalage temporel (par exemple, lorsque l'utilisateur est affecté à une tâche pendant laquelle il ne doit pas être dérangé),
- des moyens pour créer de nouveaux évènements,
- des moyens pour filtrer et supprimer des évènements.

Avec le procédé selon l'invention, on peut appliquer la séparation entre l'application et l'interface homme-machine décrite dans le brevet français FR2845174.

Selon une première variante de l'invention, le module d'interaction est réalisé avec un modèle statique, défini à la conception de la partie interaction de l'application. Ce modèle statique établit une correspondance entre les événements provenant de la vue et les appels systèmes vers le contrôleur. Cette correspondance définit le comportement de l'application en fonction des besoins de l'utilisateur. La première variante de l'invention permet un découplage entre la vue et le contrôleur. Cependant, comme ce modèle est figé à la conception du logiciel, il ne permet pas de gérer dynamiquement au cours de l'exécution de l'application, le comportement de ladite application.

Selon une deuxième variante de l'invention, le module d'interaction est réalisé avec un modèle dynamique pouvant être modifié lors de l'exécution de l'application.

Avantageusement, le modèle mis en œuvre dans le module interaction est un modèle de tâche pouvant être obtenu avec le procédé décrit dans le brevet français FR2864646. Cependant, l'interprétation sémantique pourrait potentiellement être faite par une autre méthode que le

modèle de tâche, par exemple : un modèle de l'activité ou des règles d'interprétations. En effet, le principe de l'invention est de centraliser la gestion « intelligente » de l'interaction dans un module séparé.

Un modèle de tâche décrit les différentes étapes qu'un utilisateur peut ou doit effectuer pour mener à bien sa tâche globale, son rôle ou sa mission. La tâche de l'utilisateur doit être décrite sous forme d'un enchaînement alternatif, séquentiel ou en parallèle de sous-tâches avec une mention particulière pour l'état initial. Les sous-tâches effectuées par l'utilisateur correspondent à autant d'états dans lesquels se trouve l'utilisateur (en état d'effectuer une sous-tâche).

Un modèle de tâche comprend notamment : l'enchaînement des tâches à réaliser, les événements pouvant survenir, la description des interactions à effectuer, les conditions de déclenchement nécessaires et les informations à fournir en retour. Un modèle de tâche comprend en outre un état initial et un état courant de l'utilisateur dans sa tâche permettant de suivre l'utilisateur dans sa tâche.

La figure 3 représente un exemple de modèle de tâche pour un utilisateur, client d'une compagnie aérienne, souhaitant se connecter à une application pour y consulter une liste de vols. Le modèle de tâche considéré dans l'exemple comporte deux états. Le premier état « déconnecté » 31 correspond à l'état de l'utilisateur avant sa connexion. Le second état « connecté » 32 correspond à l'état de l'utilisateur après sa connexion.

Un modèle de tâches comprend des événements permettant un changement d'état de l'utilisateur (fin d'une sous-tâche pour en débuter une autre). Ces événements peuvent être des événements à l'initiative de l'utilisateur ou des événements externes survenus dans l'environnement. L'environnement est défini ici comme tout excepté l'utilisateur : la ou les applications, les autres utilisateurs, des capteurs,... Ainsi chaque transition entre deux états de l'utilisateur est associée à une liste d'un ou de plusieurs événements déclenchant le changement d'état. Le modèle de tâche de l'exemple comporte un événement 33 correspondant à une demande de connexion de la part de l'utilisateur.

Un modèle de tâche comprend une liste de contraintes nécessaires pour déclencher une interaction. Ces contraintes peuvent être la fourniture de paramètres ou la vérification d'une valeur. Dans l'exemple, une

contrainte 34 est la fourniture de deux paramètres : un identifiant et un mot de passe.

Un modèle de tâches comprend, pour chaque événement survenu lors d'un état de l'utilisateur, une interaction à effectuer avec l'utilisateur pour gérer cet événement. L'interaction est définie comme une procédure à dérouler (interrogation d'applications, calcul d'une valeur, vérification d'une donnée,...). En fait, le modèle de tâche peut faire référence à plusieurs applications différentes. Ainsi, l'utilisateur a une seule et même interface, et ce sont alors les procédures d'interaction qui se chargent d'accéder aux applications nécessaires. Chaque procédure d'interaction doit fournir un résultat permettant de décider du prochain état de l'utilisateur. Pour chaque résultat possible de la procédure d'interaction, une transition étiquetée par le résultat permet d'atteindre un nouvel état de l'utilisateur. Dans l'exemple, la procédure d'interaction 35, après avoir interrogé l'application, fournit une réponse binaire (oui/non). Si la réponse est positive, l'utilisateur accède à l'état « connecté » 32, sinon il reste dans l'état « déconnecté » 31.

Un modèle de tâche comprend, pour chaque procédure d'interaction, les valeurs que doit fournir cette interaction selon le résultat de l'interaction et qui doivent être présentées à l'utilisateur en retour. Dans l'exemple précédent, la sortie de la procédure d'interaction permettant la connexion est la liste des vols 36 que l'utilisateur souhaite consulter.

En pratique, le premier état 31 du modèle de tâches est associé à une première vue v_1 représentant une fenêtre de connexion permettant de saisir un identifiant et un mot de passe. Le second état 32 est associé à une seconde vue v_2 présentant la liste des vols.

Avantageusement, le module d'interaction comprend un gestionnaire de tâches permettant la gestion en temps réel du modèle de tâche. Il se comporte comme un fournisseur des services décrits ci-après. Le gestionnaire est à même de gérer plusieurs modèles de tâches différents et plusieurs utilisateurs en parallèle. La multiplicité des modèles de tâches gérés par un même gestionnaire permet la collaboration entre plusieurs utilisateurs ayant des tâches potentiellement différentes. Le gestionnaire de tâches prend en compte l'évolution dans le temps d'un modèle de tâche en

offrant des services d'accès pour des modules qui pourraient altérer les modèles.

Le gestionnaire de tâches se présente sous forme d'un module externe et temps réel fournissant l'accès aux modèles de tâche dont il a la charge. Il fournit tous les services d'accès nécessaires, comme l'initialisation d'un modèle de tâche lors de la connexion d'un nouvel utilisateur, l'identification du modèle de tâche selon la classe de l'utilisateur, la fourniture de la prochaine procédure d'interaction en fonction d'un événement, la fourniture du prochain état en fonction du résultat d'une procédure d'interaction. Ce module est capable de gérer en parallèle plusieurs utilisateurs en mémorisant le contexte de chacun. La sécurité de fonctionnement de la gestion de tâche doit être assurée, par exemple avec une vérification des actions de l'utilisateur vis-à-vis de sa tâche. Le gestionnaire de tâches fournit aussi les services d'accès en écriture aux différents modèles de tâches avec sauvegarde des modèles altérés. Le gestionnaire de tâches peut utiliser un module de stockage externe afin d'accéder aux modèles de tâches des utilisateurs.

Avantageusement, le module d'interaction comprend en outre un gestionnaire de dialogue faisant l'interface entre le module d'interaction, la vue et le contrôleur. Le gestionnaire de dialogue utilise un modèle de dialogue afin de prendre en charge les enchaînements d'évènements provenant à la fois de la vue et du contrôleur.

La figure 4 montre un exemple de connexion d'un utilisateur à un système réalisé en suivant le procédé selon l'invention. Ledit système comprend un modèle (non représenté), une vue 41, un contrôleur 46 et un module interaction 45. De façon avantageuse, le module interaction comprend un gestionnaire de dialogue 42, un gestionnaire de tâches 43 gérant le modèle de tâche de l'exemple précédent stocké dans une base de données 44.

La connexion au système se déroule comme décrit ci-après. L'utilisateur saisit un identifiant de connexion et un mot de passe dans une fenêtre de la vue 41. Un premier événement e_1 « connexion » comprenant le mot de passe et l'identifiant de connexion est émis de la vue 41 vers le

module interaction 45 et plus particulièrement vers le gestionnaire de dialogue 42.

Le gestionnaire de dialogue 42 envoie alors un second événement e_2 vers le gestionnaire de tâche 43. Le second événement e_2 correspond à une requête pour connaître l'interprétation sémantique du premier événement e_1 et déterminer quel doit être le comportement de l'application. Un troisième événement e_3 est émis du gestionnaire de tâches 43 vers le gestionnaire de dialogue 42 correspondant à une réponse à la requête précédente.

Le troisième événement e_3 précise quel est l'appel vers l'application à effectuer. Dans cet exemple, il s'agit d'un appel système s_1 correspondant à une procédure de connexion. Cette procédure est externe au modèle de tâches, et seule une référence (ici son nom) permet de la retrouver. Ensuite, c'est au gestionnaire de dialogue d'appeler cette procédure. Ainsi, le modèle de tâche est totalement indépendant du contenu de cette procédure.

Un quatrième événement e_4 correspond à l'appel système s_1 de la procédure de connexion fait par le gestionnaire de dialogue 42 vers le contrôleur 46.

Le contrôleur 46 répond par un cinquième événement e_5 après l'exécution de la procédure de connexion. Cette procédure vérifie l'identifiant de l'utilisateur et son mot de passe. La procédure de connexion fournit une réponse binaire (affirmative ou négative) indiquant si l'utilisateur est autorisé à se connecter, ainsi que les paramètres 36 requis par le modèle de tâches (la liste de vol). Dans cet exemple, l'utilisateur est autorisé à se connecter à l'application.

Le gestionnaire de dialogue 42 envoie alors un sixième événement e_6 vers le gestionnaire de tâche 4. Le sixième événement e_6 est une requête pour connaître l'interprétation sémantique du cinquième événement e_5. Le gestionnaire ne fait que répondre à des requêtes et mémoriser l'état de l'utilisateur. Il mémorise donc que le nouvel état est « connecté » et crée un septième événement e_7 rendant cette réponse au gestionnaire de dialogue 42 avec les paramètres correspondant (la liste de vol).

Le gestionnaire de dialogue 42 envoie un huitième événement e_8 à la vue 41 pour lui communiquer le nouvel état de l'utilisateur et les paramètres à afficher. La vue peut alors présenter une fenêtre appropriée affichant la liste de vol.

5

REVENDICATIONS

1. Procédé de gestion de l'interaction d'au moins une application avec au moins un utilisateur, le comportement de ladite application étant dépendant du contexte d'utilisation, défini notamment par un historique de l'activité de l'utilisateur, comportant :

- une étape de création d'au moins un modèle, ledit modèle représentant de façon dynamique l'application et ses données,
- une étape de création d'au moins une vue, ladite vue présentant les données du modèle à l'utilisateur et capturant les évènements utilisateurs,
- une étape de création d'au moins un contrôleur, ledit contrôleur définissant le comportement de l'application en mettant en correspondance les évènements utilisateurs avec des altérations du modèle,

caractérisé en ce qu'il comporte en outre une étape de création d'un module d'interaction effectuant l'interprétation sémantique des évènements échangés entre la vue et le contrôleur, ladite interprétation sémantique permettant d'adapter les évènements échangés et de vérifier la cohérence entre les interactions de l'utilisateur et le contexte d'utilisation.

2. Procédé de gestion d'interactions d'une application avec au moins un utilisateur selon la revendication 1, caractérisé en ce que le module d'interaction comprend notamment :

- des moyens pour transmettre des évènements entre la vue le contrôleur,
- des moyens pour créer de nouveaux évènements,
- des moyens pour filtrer, supprimer et altérer des évènements,
- des moyens pour adapter les informations en provenance de l'utilisateur ou vers l'utilisateur.

3. Procédé de gestion d'interactions d'une application avec au moins un utilisateur selon la revendication 1 ou 2, caractérisé en ce que le module d'interaction comprend en outre des moyens pour transmettre des évènements entre la vue et le contrôleur en introduisant un décalage temporel.

4. Procédé de gestion d'interactions d'une application avec au moins un utilisateur selon l'une des revendications 1, 2 ou 3, caractérisé en ce que le module d'interaction est réalisé avec un modèle statique, défini à la conception de l'application.

5. Procédé de gestion d'interactions d'une application avec au moins un utilisateur selon l'une des revendications 1, 2 ou 3, caractérisé en ce que le module d'interaction est réalisé avec un modèle dynamique pouvant être modifié lors de l'exécution de l'application.

6. Procédé de gestion d'interactions d'une application avec au moins un utilisateur selon la revendication 5, caractérisé en ce que le module d'interaction réalise l'interprétation sémantique des événements à l'aide d'un modèle de tâche, ledit modèle de tâche comprenant notamment :

- un enchaînement de tâches à réaliser par l'utilisateur,
- des événements pouvant survenir,
- une description des interactions à effectuer lorsqu'un événement survient,
- les conditions de déclenchement nécessaires desdites interactions,
- les informations à fournir en retour par lesdites interactions,
- un état courant de l'utilisateur dans sa tâche.

7. Procédé de gestion d'interactions d'une application avec au moins un utilisateur selon la revendication 6, caractérisé en ce que le module d'interaction comprend : un gestionnaire de tâche pour gérer en temps réel les modèles de tâches, une base de donnée pour stocker les modèles de tâches.

8. Procédé de gestion d'interactions d'une application avec au moins un utilisateur selon la revendication 7, caractérisé en ce que le module d'interaction comprend en outre un gestionnaire de dialogue faisant l'interface entre le module d'interaction, la vue et le contrôleur.

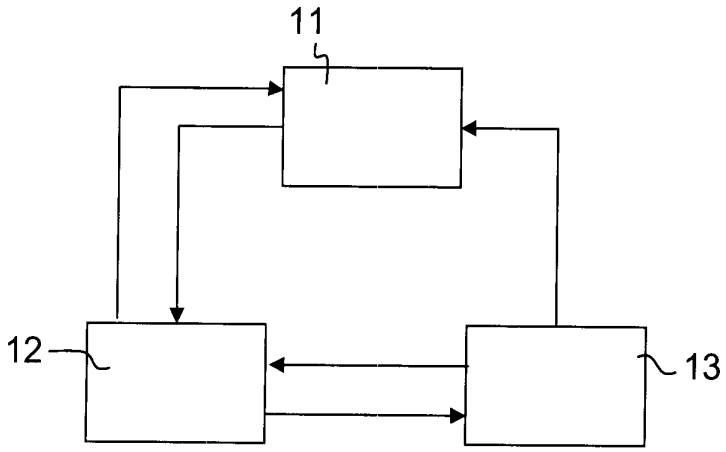


FIG.1

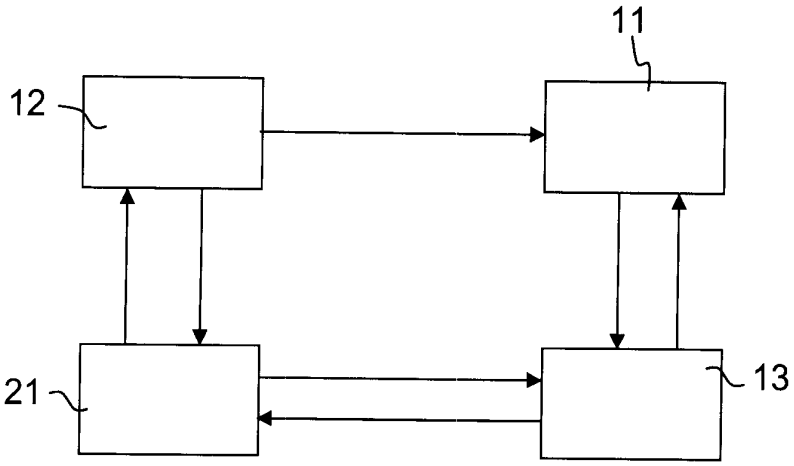


FIG.2

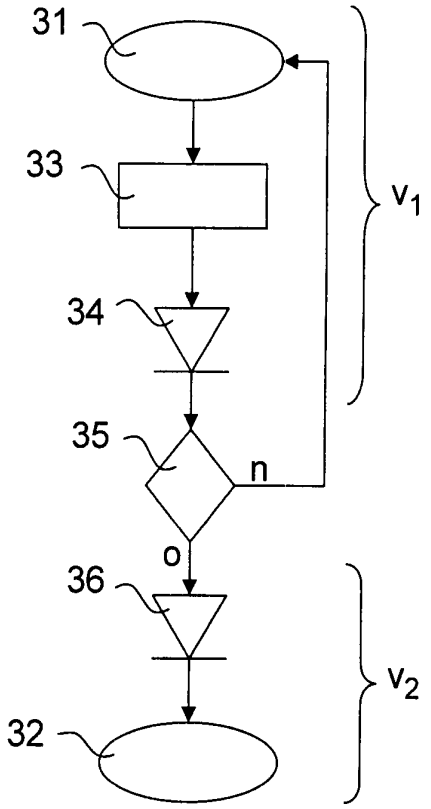


FIG.3

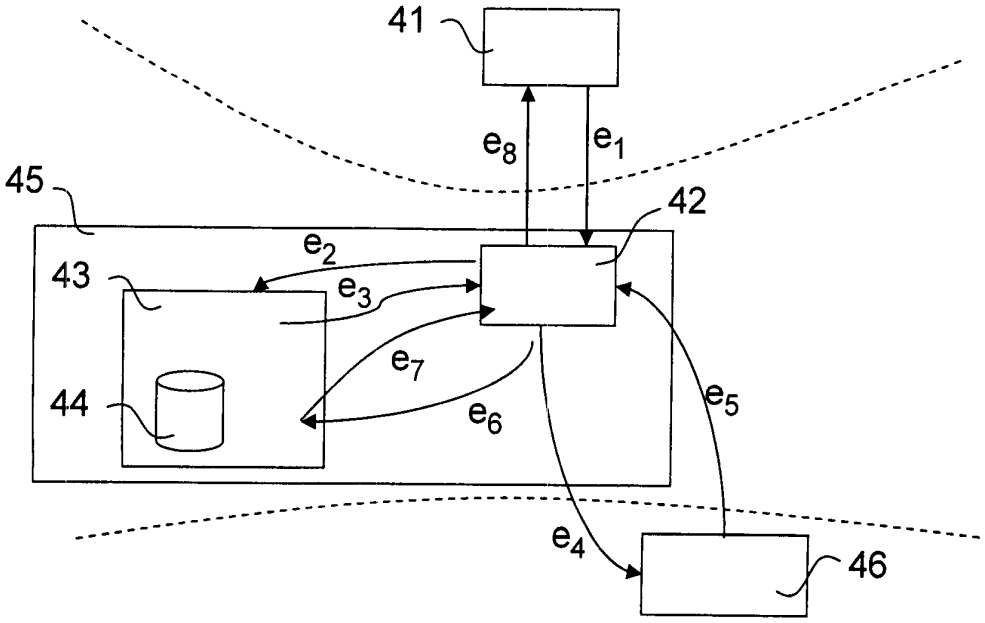


FIG.4

RÉPUBLIQUE FRANÇAISE

2913509

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

N° d'enregistrement national

FA 691532
FR 0701622

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
X	US 2003/023953 A1 (LUCASSEN JOHN M [US] ET AL) 30 janvier 2003 (2003-01-30) * abrégé * * alinéa [0002] * * alinéas [0004], [0005] * * alinéas [0010] - [0015] * * alinéa [0028] * * alinéas [0033] - [0038] * * alinéas [0044], [0045] * * alinéas [0059], [0060] * * alinéas [0104] - [0109] * * alinéas [0115] - [0121] * * revendications 16-25 * -----	1-8	G06F9/44 G06F19/00 G06Q90/00
X	EP 0 822 484 A2 (SUN MICROSYSTEMS INC [US]) 4 février 1998 (1998-02-04) * le document en entier * -----	1-8	
A	ANDERSON D J: "Using MVC Pattern in Web Interactions" INTERNET CITATION, [Online] 22 juillet 2000 (2000-07-22), XP002310590 Extrait de l'Internet: URL:http://www.uidesign.net/Articles/Papers/WebMVC-BrowsersTransactio.html> [extrait le 2004-12-14] * le document en entier * -----	1-8	DOMAINES TECHNIQUES RECHERCHÉS (IPC) G06F
A	EP 1 594 279 A (HEWLETT PACKARD DEVELOPMENT CO [US]) 9 novembre 2005 (2005-11-09) * abrégé * * alinéa [0008] * * alinéas [0016] - [0023] * * revendications 1-11 * -----	1-8	

Date d'achèvement de la recherche	Examinateur
7 août 2007	TOMAS BLANCH, F

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)

ANNEXE AU RAPPORT DE RECHERCHE PRÉLIMINAIRE RELATIF A LA DEMANDE DE BREVET FRANÇAIS NO. FR 0701622 FA 691532

La présente annexe indique les membres de la famille de brevets relatifs aux documents brevets cités dans le rapport de recherche préliminaire visé ci-dessus.
Les dits membres sont contenus au fichier informatique de l'Office européen des brevets à la date du 07-08-2007
Les renseignements fournis sont donnés à titre indicatif et n'engagent pas la responsabilité de l'Office européen des brevets, ni de l'Administration française

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 2003023953 A1	30-01-2003	US 2005273759 A1	08-12-2005
EP 0822484 A2	04-02-1998	DE 69734545 D1	15-12-2005
		DE 69734545 T2	20-07-2006
		JP 11003237 A	06-01-1999
		US 5999728 A	07-12-1999
EP 1594279 A	09-11-2005	US 2005262099 A1	24-11-2005

EPO FORM P0465