La présente invention concerne le domaine des architectures informatiques, et en particulier un procédé pour le développement d'applications destiné à un fonctionnement en réseau, par exemple sur un le réseau Internet.

L'invention se situe dans le domaine de la programmation graphique par composants, permettant à des développeurs de d'utiliser des objets informatiques autonomes, et de gérer leurs interactions à l'aide d'outils de développement mettant en œuvre une interface graphique permettant d'automatiser la création des procédures et des fichiers de communication inter-objets.

Un certain nombre de techniques proposent aujourd'hui un concept de programmation graphique à base de modules indépendants :

- on dispose d'un certain nombre de briques (ou modules) qui, comme un circuit intégré, présentent des entrées et des sorties nommées
- chaque module n'a aucune connaissance des autres
- le développement d'une application consiste à assembler un nombre quelconque de modules en reliant des sorties à des entrées

ce principe s'accorde bien avec une représentation graphique, on trouve donc des outils qui permettent à l'utilisateur de placer des modules sur une " feuille ", puis de tracer des flèches entre ces modules, d'une sortie vers une entrée.

De plus, certaines technologies présentent un concept de distribution et permettent donc de répartir les modules sur un ensemble de machines. Dans ce cas, le module reste indivisible et est placé entièrement sur une machine.

Le but de l'invention est d'étendre ces techniques à des sites Internet dans lesquels les visiteurs vont pouvoir interagir. Dans un tel site, le traitement est réparti entre le serveur et les clients.

Le serveur est le point de connexion du site : c'est le point auquel on se connecte pour entrer dans le site. Il est sous le contrôle du propriétaire du site. Les clients sont les utilisateurs qui se connectent à distance sur le site : leur machine n'est pas sous le contrôle du propriétaire du site, et cependant elle va participer au fonctionnement du site en effectuant une partie des traitements.

La solution classique qui consiste à distribuer des modules trouve ici des limites :

- pour un système de saisie de mot de passe :
- le client doit afficher l'interface de saisie, crypter le mot de passe et le transmettre au serveur
- le serveur reçoit le mot de passe, le décrypte et le vérifie
- pour un système de discussion en environnement 3d :
- le client calcule et affiche un environnement en 3 dimensions
- le serveur informe chaque client de la présence des autres et sert éventuellement de relai et de filtre dans les discussions.

Chaque fonctionnalité devrait donc être découpée en deux objets hétérogènes : un objet fonctionnant sur le serveur, et un objet fonctionnant sur les clients.

La communication qui résulte entre ces deux composantes d'une même fonctionnalité deviendrait alors visible dans l'outil de création de site, puisqu'elle serait représentée par des liens qui ne présenteraient aucune souplesse (il y aurait obligation de les tracer).

On rappelle également les deux principes de communication :

- principe de sélection : à qui veut-on envoyer un message ?
- principe de traduction : que veut-on signifier, et comment le faire comprendre ?

On connaît dans l'état de la technique différents brevets concernant de telles architectures.

Par exemple, le brevet européen EP732834 décrit un système et une méthode pour la détermination et la manipulation de l'information de configuration des serveurs dans un environnement d'objets répartis.

Le brevet européen EP546683 décrit un système de traitement de données orienté objet présentant un moyen destiné à produire un module objet chargeable à partir d'une définition d'interface d'objets indépendants d'un langage, comprenant un moyen de mémorisation destiné à mémoriser la définition d'interface d'objets indépendants d'un langage. Les liens de langage comprennent une structure de données de classes externe aux informations binaires, et un moyen destiné à compiler les liens de langage afin de produire un module objet chargeable.

On connaît également le brevet américain US5915113 qui décrit un procédé de programmation-objet qui permet de répartir des objets informatiques et de créer des liens entre eux.

Le brevet américain US5724589 décrit un procédé de programmation graphique dans lequel les objets informatiques sont dépourvus de connaissances mutuelles.

Le brevet américain US5991535 décrit un autre procédé de programmation objet.

Les procédés de l'art antérieur concernent tous des procédés où les objets sont chargés et exécutés soit sur le serveur, soit sur un poste client. Cela implique plusieurs inconvénients. Tout d'abord, il n'est pas possible d'optimiser les ressources informatiques client ou serveur, car les objets sont nécessairement exécutés sur la machine sur laquelle ils sont chargés. D'autre part, le poste client accède aux objets au moment de leur exécution, et il n'est donc pas possible de préserver la confidentialité de ces objets.

Le but de l'invention est de remédier à ces inconvénients en proposant un procédé mettant en œuvre des objets

qui sont divisibles dynamiquement en deux ou plusieurs modules complémentaires, qui se créent et se détruisent sur le poste client de façon dynamique.

Les modules complémentaires d'un même objet dialoguent entre eux par des messages circulant sur le réseau.

L'invention concerne dans son acception la plus générale un procédé pour la programmation graphique consistant à mémoriser, pendant la phase d'exécution, dans au moins une mémoire d'un serveur et dans la mémoire d'un ou de plusieurs clients des objets informatiques autonomes comprenant chacun des interfaces d'entrée et des interfaces de sortie pour communiquer entre eux, la liaison entre les interfaces d'entrée et de sortie étant établie par un module informatique générant les liaisons à partir d'une saisie graphique caractérisé en ce que certains objets au moins sont constitués par un module informatique autonome constitué de deux composantes, chacune desdites composantes étant indivisible et apte à échanger des informations formées par des données numériques avec la composante complémentaire, l'une desdites composantes étant chargée dans la mémoire vive du serveur et l'autre desdites composantes étant chargée dans la mémoire vive du client, et en ce qu'un module est dépourvu de moyens pour accéder à un autre module.

Chaque objet peut avoir un fonctionnement distribué : une partie du traitement s'effectue sur le serveur, et une autre partie du traitement s'effectue sur le client.

Selon une variante, l'ensemble des objets informatiques et des modules sont mémorisés, en dehors des phases d'exécution, uniquement dans la mémoire du serveur.

Avantageusement, les objets constitués de deux modules informatiques complémentaires comportent un module transféré, au moment de leur exécution, dans la mémoire du poste client et un module complémentaire enregistré seulement dans la mémoire du serveur, les deux modules étant aptes à échanger des données

numériques par l'intermédiaire d'un réseau sur lequel sont connectés le client et le serveur.

Selon un mode de réalisation préféré, le module enregistré dans la mémoire du client est activé par des instructions transmises par le module enregistré dans la mémoire serveur.

De préférence, les modules téléchargés par le client sont des fichiers exécutables, écrits dans un quelconque langage informatique.

Selon une variante, le module serveur comporte des moyens pour permettre l'échange de données numériques entre deux modules téléchargés par le client.

Selon un mode de mise en œuvre particulier, les objets informatiques sont enregistrés, avant leur exécution, dans plusieurs serveurs distincts reliés entre eux par un réseau informatique.

Avantageusement, les données informatiques échangées entre le ou les serveurs, et le ou les clients sont constitués par un message comportant un identifiant de client et éventuellement un paramètre.

De préférence, le chargement en mémoire vive d'un module sur le poste client est commandé par la réception d'un message comportant l'identifiant dudit module.

L'invention sera mieux comprise à la lecture de la description qui suit, se référant à des exemples de mise en œuvre non limitatifs.

Le procédé selon l'invention est fondé sur le principe qu'une fonctionnalité doit correspondre à un seul objet, car il est plus facile à un non-technicien de manipuler des fonctionnalités complètes plutôt que des bouts de fonctionnalités.

Chaque objet se présente donc comme une boîte possédant des entrées et des sorties.

Chaque fonctionnalité se décompose de manière interne en au plus deux parties :

- une partie fonctionnant sur le serveur
- une partie fonctionnant éventuellement sur les postes clients (celle-ci n'est pas toujours requise)

Cela pose la question de la localisation des entrées et des sorties : chacune peut être localisée sur la partie serveur ou sur la partie cliente.

On établit une distinction entre :

- la communication intra-objet : une partie cliente d'un objet communique avec la partie serveur du même module, ou inversement.
- la communication intra-objet : un module communique avec un autre

Communication intra-objet.

Le principe de traduction est le suivant : les parties serveur et client d'un même objet sont sensées se connaître et donc parler le même langage.

Le principe de sélection est simple dans le sens client vers serveur puiqu'il n'y a qu'une seule partie serveur pour un module.

Dans le sens serveur vers client, le principe de sélection implique de choisir vers quel module client envoyer le message.

Communication inter-module

Le principe de traduction est résolu par les liens : chaque lien relie une sortie et une entrée nommées, et définit donc une correspondance entre deux noms, ce qui est le propre de la traduction.

Le principe de sélection est partiellement résolu par les liens : chaque lien définit un objet émetteur et un objet récepteur.

Il faut encore résoudre le problème de localisation de l'entrée :

- si l'entrée est sur la partie serveur de l'objet récepteur, il n'y a pas de question puisque la partie serveur est unique
- si l'entrée est sur la partie client de l'objet récepteur, il faut préciser vers quel client le message doit être transmis.

Graphe User-Flow

Il en résulte que les messages doivent contenir une information supplémentaire, qui se révèle être l'information fondamentale, à savoir l'indication d'un client.

Le site est donc constitué d'un graphe statique de modules, dont le principe mobile est les clients.

On généralise les clients à la notion d'utilisateur (user). On associe un unique 'user' à chaque client et on distingue :

- les 'users' qui correspondent à un client
- les 'users' qui ne correspondent à aucun client et qui résident en permanence dans le serveur.

On ne s'interdit pas de faire passer un 'user' d'un état à l'autre : le 'user' d'un client reste dans le système après déconnexion, et est réattribué au client lorsque celui-ci revient.

On définit un message inter-objet comme étant :

- un 'user', qui est le user concerné directement par le message, et qui sert à résoudre les questions de routage
- éventuellement un paramètre quelconque, permettant de gérer des flux de données entre modules et pas seulement des signaux
- éventuellement une liste de 'users' indirectement concernés par l'action
- éventuellement une fonction de réponse

On autorise tout objet à émettre de tels messages sur ses sorties. L'architecture selon l'invention prend en charge ce message et le transmet en suivant les liens du graphe.

On définit deux types d'entrées de module :

- une entrée serveur : l'objet serveur prend en charge les messages qui arrivent sur cette entrée
- une entrée client : l'objet client prend normalement en charge les messages qui arrivent sur cette entrée. Le serveur peut également proposer de prendre en charge une telle entrée pour les 'users' qui ne correspondent pas à un client, ainsi que pour les 'users' correspondant à un client mais pour lesquels le module client n'a pas été activé

On définit les règles de routage :

- message émis par un module serveur
- message se dirigeant vers une partie serveur :
  - pas de question particulière de routage
- message se dirigeant vers une partie cliente :
  - soit le 'user' correspond à un client, et alors le message est envoyé à ce client
  - soit le 'user' ne correspond pas à un client, et alors le message est quand même transmis au serveur.
- message émis par un module client
- Un tel message référence obligatoirement le 'user' associé au client.
- message se dirigeant vers une partie serveur :
- pas de question particulière de routage
- message se dirigeant vers une partie cliente :
- le message est transmis localement

Activation dynamique

De par la structure décrite précédemment, il résulte que le même graphe est utilisé côté serveur et côté client. On parlera d'architecture à copie. Cependant cette copie est partielle et dynamique : pour des raisons d'efficacité et de sécurité, il n'est pas utile que les modules clients soient tous en fonctionnement à un instant donné chez un client donné.

Certaines parties du site peuvent en effet rester inaccessibles à un client.

Le principe d'activation dynamique consiste à laisser chaque module serveur décider de l'activation du module client correspondant : il décide lui-même d'activer/désactiver la partie cliente chez un client donné. C'est à ce moment seulement que le client prendra connaissance de cette partie du graphe.

Mise en œuvre du procédé

Le procédé selon l'invention permet la création de sites Internet. Le concepteur du site définit une architecture par la sélection d'objets pré-existant et par l'établissement de liens entre ces objets. Chaque objet gère une ou plusieurs fonctionnalités : log, authentification, espace 3D, … Chaque objet peut avoir un fonctionnement distribué : une partie du traitement s'effectue sur le serveur, et une autre partie du traitement s'effectue sur le client.

Chaque objet est divisé en deux partie complémentaires dont l'une fonctionne sur le serveur et l'autre sur les clients. La communication entre la partie client et la partie serveur d'un même objet sera gérée exclusivement par l'auteur du module. La communication entre deux modules se fait exclusivement sous la forme de liens. Les liens sont directionnels : de l'événement vers l'action. A tout moment, un objet peut déclencher un événement. Le système le convertit, selon les liens, en actions pour d'autres modules. Le module concerné par l'action est averti de l'identité de l'émetteur, et peut lui répondre, en utilisant la fonction de réponse attachée au message.

Un éditeur de site permet de :

- sélectionner les modules à intégrer au site
- définir les documents serveur et client
- définir les liens entre les objets
- affecter les zones au modules : indiquer dans quelle zone afficher la 3D, un bouton, une image

- lancer les éditeurs de modules. Chaque module contient un éditeur qui permet de paramétrer le module (définir un espace 3D pour un module de gestion 3D, définir les textes d'un module " bandeau d'affichage ",…)

Le serveur joue le rôle de distributeur de ressources. Une ressource correspond à un ensemble de données numériques. Elle appartient à un objet qui est son propriétaire. Les ressources sont les données que les clients vont pouvoir télécharger : fichier contenant le programme du module client, fichier graphique, 3D, son, …

Chaque module enregistre les ressources dont le client peut avoir besoin, afin de garantir les possibilités de mise à jour automatique. Une ressource est normalement accessible à tout client dont le module client correspondant au module propriétaire est activé. Cependant, la ressource peut être sécurisée avec un accès restreint aux seuls clients autorisés.

Coté serveur, la ressource peut être présente dans la mémoire du serveur ou stochée sur le disque. Dans le premier cas seulement, elle pourra être compressée avant le téléchargement par le client. Dans le deuxième cas, les données seront transférées directement : il est donc recommandé que le fichier soit déjà compressé (fichier graphique jpeg par exemple).

Coté client, la ressource peut être stockée dans le cache sous un nom donné qui permettra, lors d'un usage ultérieur, de déterminer si la ressource doit être téléchargée ou si elle est présente. La ressource peut également être fournie directement au client, sans passer par un stockage sur un disque.

Le serveur calcule la signature des ressources qu'on lui soumet. Cette signature est transmise au module client lors de sa création. Le module client vérifie la signature du fichier portant le nom de la ressource et si elle correspond, il télécharge de nouveau le module.

Le serveur peut donc :

- enregistrer une ressource en spécifiant son nom et son propriétaire
- désenregistrer une ressource en spécifiant son nom
- désenregistrer toutes les ressources d'un propriétaire donné
- autoriser un client à accéder à une ressource.

Le client peut :

- demander le téléchargement d'une ressource en connaissant son nom et en précisant le nim du module demandeur, en passant par le cache ou non
- arrêter le teléchargement d'une ressource
- arrêter tous les teléchargements commandé par un module donné.

Revendications

1 – Procédé pour la programmation graphique consistant à mémoriser, pendant la phase d'exécution, dans au moins une mémoire d'un serveur et dans la mémoire d'un ou de plusieurs client des objets informatiques autonomes comprenant chacun des interfaces d'entrée et des interfaces de sortie pour communiquer entre eux, la liaison entre les interfaces d'entrée et de sortie étant établie par un module informatique générant les liaisons à partir d'une saisie graphique caractérisé en ce que certains objets au moins sont constitués par un module informatique autonome constitué de deux composantes, chacune desdites composantes étant indivisibles et aptes à échanger des informations formées par des données numériques avec la composante complémentaire, l'une desdites composantes étant chargée dans la mémoire vive du serveur et l'autre desdites composantes étant chargée dans la mémoire vive du client, et en ce qu'un module est dépourvu de moyens pour accéder à un autre module.

2 - Procédé pour la programmation graphique selon la revendication 1 caractérisé en ce que l'ensemble des objets informatiques et des modules sont mémorisés, en dehors des phases d'exécution, uniquement dans la mémoire du serveur.

3 - Procédé pour la programmation graphique selon la revendication 1 ou 2 caractérisé en ce que les objets constitués de deux modules informatiques complémentaires comportent un module transféré, au moment de leur exécution, dans la mémoire du poste client et un module complémentaire enregistré seulement dans la mémoire du serveur, les deux modules étant aptes à échanger des données numériques par l'intermédiaire d'un réseau sur lequel sont connectés le client et le serveur.

4 - Procédé pour la programmation graphique selon la revendication 3 caractérisé en ce que le module enregistré dans la mémoire du client est activé par des instructions transmises par le module enregistré dans la mémoire serveur.

5 - Procédé pour la programmation graphique selon l'une au moins des revendications précédentes caractérisé en ce que les modules téléchargés par le client sont des fichiers exécutables, écrits dans un quelconque langage informatique.

6 - Procédé pour la programmation graphique selon l'une au moins des revendications précédentes caractérisé en ce que le module serveur comporte des moyens pour permettre l'échange de données numériques entre deux modules téléchargés par le client.

7 - Procédé pour la programmation graphique selon l'une au moins des revendications précédentes caractérisé en ce que les objets informatiques sont enregistrés, avant leur exécution, dans plusieurs serveurs distincts reliés entre eux par un réseau informatique.

8 - Procédé pour la programmation graphique selon l'une au moins des revendications précédentes caractérisé en ce que les données informatiques échangées entre le ou les serveurs, et le ou les clients sont constituées par un message comportant un identifiant d'un client et éventuellement un paramètre.

9 - Procédé pour la programmation graphique selon la revendication précédente caractérisé en ce que le chargement en mémoire vive d'un module sur le poste client par la réception d'un message comportant l'identifiant dudit module.

10 - Procédé pour la programmation graphique selon l'une au moins des revendications précédentes caractérisé en ce

qu’un objet est constitué par des fichiers informatiques pour l’exécution d’une fonctionnalité complète.