Proposition de sujet de Thèse pour Monsieur Benoît Sonntag
Directeur de Thèse: Dominique Colnet, Maître de Conférences à l'UHP.
Equipe ECOO – Environnements Coopératifs –
Responsable d'équipe: Claude Godart, Professeur à l'ESSTIN.

Intégration de concepts objets au coeur même des systèmes d'exploitation

Mots-clés: concepts objets, noyau de système d'exploitation, compilation, inférence de types, contrôle de cohérence, optimisation, parallélisme, distribution.

L'évolution des technologies et du matériel nous amène à revoir la conception du noyau des systèmes d'exploitation actuels. Nous pensons que l'introduction de concepts objets comme l'héritage et la liaison dynamique, au coeur même des systèmes d'exploitation, peut simplifier et optimiser l'utilisation des divers composants physiques d'un ordinateur. Il est essentiel de pouvoir s'adapter dynamiquement à la diversité des périphériques tout en restant compact, comme par exemple dans les systèmes embarqués. De manière générale, notre démarche consiste à repenser la définition d'un noyau système en utilisant des concepts de programmation de haut niveau: intégration de concepts objets, gestion mémoire automatique, problèmes de synchronisation, distribution sur plusieurs processeurs et virtualisation de l'architecture.

Aspects dynamiques

Qu'il s'agisse d'un langage à base de classes ou de prototypes, l'aspect dynamique des langages à objets est selon nous un outil de base indispensable pour le concepteur de système d'exploitation. Dans les noyaux systèmes actuels, l'aspect dynamique des langages à objets, bien que conceptuellement présent, reste à l'état embryonnaire. Nous pensons être en mesure de franchir le pas (ie. définir le noyau à l'aide d'un langage de haut niveau) sans remettre en cause l'efficacité des services offert par le noyau. Le gain attendu d'une telle démarche consiste dans la simplification et dans la sécurisation du noyau lui-même ainsi qu'une amélioration des services rendus en terme de niveau d'abstraction.

D'un point de vue mise en oeuvre, les langages à objets sont passés par différent stades de développement, de l'interprétation à la compilation, à base de classes ou de prototypes, partiellement ou complètement dynamiques. Une des premières étapes de notre travail va consister à recenser quels sont les aspects indispensables pour le langage de définition du noyau système. Faut il utiliser des classes ou des prototypes? Héritage simple ou héritage multiple? Programmation par contrats? Faut il considérer un processus comme une objet ou non? etc.

Le système se composerait de petits programmes exécutables, que l'on assimilerait à des objets, offrant un certain nombre de services sur un domaine précis. Plusieurs techniques de communication, de modification et d'enrichissement des objets sont à envisager:

- Chaque objet pourra communiquer par des services sous forme de fonctions, procédures ou attributs. Nous pouvons imaginer d'affiner leurs droits d'accès, par famille d'objets ou par niveau de privilège.
- Un objet pourra être à l'origine de différentes déclinaisons de ces méthodes, par héritage simple ou multiple. Cela permet de faire évoluer un objet indépendamment de toute sa descendance.

Voici donc les deux axes principaux du concept d'objet dynamique, que nous pourrions étendre aux concepts objets du langage Eiffel.

Bien évidemment pour tirer pleinement partie de la modularité de l'objet nous étendons le concept au niveau applicatif. C'est à dire qu'à l'image du noyau, un logiciel devient lui-même un assemblage d'objets permettant une adéquation parfaite entre les besoins et le produit qui devient adaptatif par insertion ou suppression d'objets. Cette approche permet une complémentarité des

compétences des développeurs dans différents domaines applicatifs.

Vers une architecture virtuelle

Nous savons qu'il existe une équivalence d'expression, en terme de calculabilité, entre les réseaux de Pétri et la machine de Turïng. Nous voudrions utiliser la puissance des algorithmes liés aux graphes pour s'adapter à toute architecture. Nous pouvons imaginer de travailler sur des graphes orientés, dont les noeuds représentent des instructions élémentaires, et les arcs leurs paramètres et leur ordonnancement en séquence. La méthode consisterait donc, dans un premier temps, à connaître le graphe équivalent à chaque instruction d'une architecture. Nous compilerions donc les objets en un graphe constitué d'instructions élémentaires indépendantes de l'architecture. L'installation d'un nouvel objet dans un système effectuera une traduction de son graphe en une suite d'instructions du processeur. Cette transformation utilisera les algorithmes de matching pour trouver les correspondances entre les graphes d'instructions et le graphe de l'objet. Par cette méthode le code s'adapte automatiquement au mieux à l'architecture. Par ailleurs, il serait intéressant de réfléchir aux propriétés des graphes qui permettraient de paralléliser le code. En effet, le code sous forme de graphe permet de travailler sur les dépendances des instructions permettant ainsi une répartition du code parmi les différentes unités du (ou des) processeur(s).

Une exécution répartie

Avec une telle segmentation du code exécutable, nous pouvons facilement imaginer une meilleure utilisation des ressources d'un parc informatique. En effet, une machine peu sollicitée pourrait être utilisée pour l'exécution d'objet distant, libérant ainsi la charge des autres machines, d'où une meilleure répartition des coûts. Cela demanderait la réalisation d'un ordonnancement global des ressources et une communication accrue des machines entre elles.

L'objet de cette thèse permettra donc d'étendre et de concrétiser des concepts intéressants et novateurs dans le domaine des systèmes d'exploitation. Nous sommes conscient du caractère ambitieux de ce projet qui doit être validé par la mise en oeuvre d'un prototype capable d'intégrer des services comparables à ceux offert par Linux.