MARC SAINT-LAURENT

Plan de projet

Environnement graphique pour la spécification de propriétés de systèmes informatiques

Travail présenté à M. Michal Iglewski, Ph.D.
sous la supervision de M. Kamel Adi, Ph.D.
dans le cadre du cours INF4173 – Projet synthèse

Département d'informatique et d'ingénierie
Université du Québec en Outaouais
12 mai 2009

DESCRIPTION

Le projet consiste principalement à développer une application qui génère les spécifications d'une topologie de réseau. Plus précisément, l'utilisateur de cette application doit être en mesure de créer un modèle de topologie à partir d'éléments graphiques qui représentent des ponts, pare-feu, routeurs, stations de travail, etc. De plus, l'outil doit permettre de créer et modifier les liens entre les divers éléments du réseau. Le programme doit aussi afficher les propriétés de tous ces objets. Finalement, il doit savoir traduire la représentation graphique de la topologie en une expression formelle dans une algèbre de processus.

CONTEXTE

Cette application sera réalisée pour enrichir / augmenter les résultats de recherche du groupe LRSI (Laboratoire de Recherche en Sécurité Informatique). Elle pourrait constituer le point de départ d'un projet plus complexe de développement d'un environnement pour le renforcement automatique de politiques de sécurité dans les systèmes informatiques.

Un cadre formel de spécification et vérification des systèmes informatiques a été conçu par des membres du groupe LRSI. La spécification se fait à l'aide d'une nouvelle algèbre basée sur le calcul ambiant (*ambient calculus*). Elle permet de décrire le comportement des divers composants d'un réseau, incluant ses politiques de sécurité. La spécification peut servir à confirmer la cohérence d'une politique de sécurité réseau, ou même de trouver comment un intrus pourrait exploiter ou contourner l'implémentation défectueuse de certaines politiques.

Le logiciel qui sera créé dans le cadre de ce projet servira à créer la représentation graphique d'une topologie de réseau. L'utilisateur pourra affecter des processus aux divers nœuds du réseau. Finalement, l'application devra être en mesure de générer la spécification qui décrit la topologie.

OBJECTIFS

À la fin de ce projet, le LRSI doit disposer d'un outil graphique qui permet de :

- Modéliser une topologie de réseau en glissant les éléments de réseau dans une fenêtre,
- Définir les liens entre les nœuds du réseau,
- Visualiser les propriétés de tous les composants du réseau (nom, identifiant, processus, etc.),
- De créer une expression formelle qui décrit la topologie.

En ce qui concerne l'interface homme-machine, elle doit être conviviale et facile d'utilisation. Il faut pouvoir manipuler les éléments du modèle de réseau avec la souris. Il est préférable de permettre l'utilisateur de déplacer les objets. L'application sera alors responsable de redessiner les liens vers l'objet déplacé. Afin de faciliter son utilisation, le programme doit suivre les standards d'utilisation bien connus (par exemple : Ctrl-C pour copier un nœud, Ctrl-V pour coller une copie du nœud, etc.).

MÉTHODOLOGIES

D'après les spécifications du projet, le programme doit être écrit avec le langage Java. La partie graphique de l'application sera développée à l'aide des librairies Swing/JFC, puisqu'elles sont très bien documentées et font déjà partie de tous les environnements Java récents.

Certaines parties du programme devront être bien testés pour confirmer leur bon fonctionnement (la génération de la spécification est le cas le plus évident). L'utilisation d'un cadre tel que Junit est envisagé afin de faciliter la création et le lancement des tests unitaires.

L'utilisation de l'outil Ant de Apache est aussi prévue. Il permet d'automatiser la grande majorité des tâches du programmeur, tout au long du cycle de développement. Entre autres, Ant servira à compiler les sources, à faire le nettoyage avant la compilation, à la génération des données de tests, et à lancer les tests.

Le programme sera développé en suivant des patrons conception bien connus, tels que MVC (Modèle – Vue – Contrôleur), et/ou le patron de l'observateur. Ces modèles d'architecture permettent de bien séparer l'interface de l'implémentation. La raison sous-tendante de ce choix est que les politiques de sécurité ne s'appliquent pas uniquement aux réseaux. Si un jour on décide d'étendre la fonctionnalité de ce programme, le développeur pourra plus aisément le modifier si les divers modules ne sont pas fortement couplés.