ETH zürich

ETH Library

Separate compilation and module extension

Doctoral Thesis

Author(s):
Crelier, Régis Bernard Joseph

Publication date:
1994

Permanent link:
https://doi.org/https://doi.org/10.3929/ethz-a-000945227

Rights / license:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

Diss. ETH ex.B

Diss. ETH No 10650

Separate Compilation and Module Extension

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of
Doctor of Technical Sciences

presented by
Régis Bernard Joseph Crelier
ing. informaticien dipl. EPFZ
born January 10, 1965
citizen of Bure, Jura

CaIE

accepted on the recommendation of
Prof. Dr. N. Wirth, examiner
Prof. Dr. H. Mössenböck, co-examiner

1994

Résumé

L'évolution incessante du matériel informatique permet la construction d'ordinateurs toujours plus performants. Parallèlement, de nouveaux concepts et de nouvelles techniques de programmation deviennent nécessaires afin de maîtriser la complexité grandissante des logiciels. L'une de ces techniques, la compilation séparée de modules, a fait ses preuves en Modula-2 et en Obéron, entre autres langages de programmation fortement typés.

Le module, qui est à la fois l'unité structurelle et l'unité de compilation des programmes, est interchangeable sans affecter les autres modules d'un système. Toutefois, si l'interface du nouveau module est différente de l'ancienne, les modules clients de cette interface modifiée doivent être recompilés afin de maintenir la cohérence du système. L'éditeur de liens vérifie habituellement que la clef réelle de chaque interface importée est identique à la clef attendue par le client, une différence indiquant une incohérence. Ce modèle simple et efficace n'est pas très souple, puisqu'une légère modification d'interface, telle l'insertion d'une nouvelle procédure, par exemple, peut provoquer de nombreuses recompilations inutiles.

Cette thèse présente deux nouveaux modèles de vérification fine de cohérence, ainsi que leur mise en oeuvre. Ces modèles permettent l'extension d'interfaces de modules compilés séparément sans qu'une recompilation des clients ne soit nécessaire, ce qui est particulièrement utile dans des systèmes avec chargement dynamique, où les clients d'une bibliothèque de modules ne sont pas forcément connus lors de la révision ou de l'extension de cette bibliothèque. L'édition d'interface sans conséquence ne se limite pas à des extensions, puisque la modification d'un article existant n'invalide pas les clients n'utilisant pas cet article. Et même s'ils l'utilisent d'une manière encore compatible après modification, une recompilation n'est pas plus indispensable.

Ces techniques ont été implantées dans le Système Obéron, mais elles ne sont spécifiques ni au Langage, ni au Système Obéron, et pourraient être appliquées à n'importe quel système de programmation modulaire pour en améliorer la sécurité, la flexibilité, ou tous les deux à la fois. De plus, ces techniques ne se présentent pas au programmeur sous forme d'outils dont l'utilisation reste facultative, mais elles sont entièrement intégrées au compilateur ainsi qu'au chargeur de modules. La sécurité ne doit pas être offerte en option.

Abstract

As continuous evolution in hardware results in more powerful computers, new programming techniques and concepts must be developed to master the consequently increasing software complexity. Separate compilation of modules is such a technique that has proven valuable in Modula-2 and in Oberon, among other strongly-typed programming languages.

The module is both the structural unit and the compilation unit of programs. Replacing a module by a new one does not affect the rest of the system, provided that the module interface has not changed. Otherwise, client modules of the modified interface have to be recompiled to maintain system consistency. The last opportunity to detect an inconsistency is when modules are linked to form an executable unit. The check usually consists in comparing, for each imported interface, the expected key of that interface, as known at compilation time of the client, with the key of the effectively supplied interface. A mismatch indicates an inconsistency. This model is simple and efficient, but not very flexible. Indeed, a minor modification of an interface, such as the insertion of a new procedure, can trigger many unnecessary recompilations.

This thesis presents two new models for fine-grained consistency checking and their implementation. These models allow the interface of separately compiled modules to be extended without requiring a recompilation of client modules. This is particularly valuable in systems with dynamic loading, where the clients of a library are not known when the library is revised or extended. Interface editing that does not require client recompilation is not restricted to extensions, since the modification of an existing item does not invalidate clients not using this particular item. Even if they use it in a way that is upward-compatible with the modification, they still do not need a recompilation.

These techniques have been implemented in the Oberon System, but they are neither specific to the Oberon Language nor to the Oberon System. They can be applied to any modular programming system in order to improve its safety, its flexibility, or both. Furthermore, these techniques are not available to the programmer as a separate tool whose use remains optional, but have been fully and transparently integrated into the compiler and module loader. Safety must not be optional.