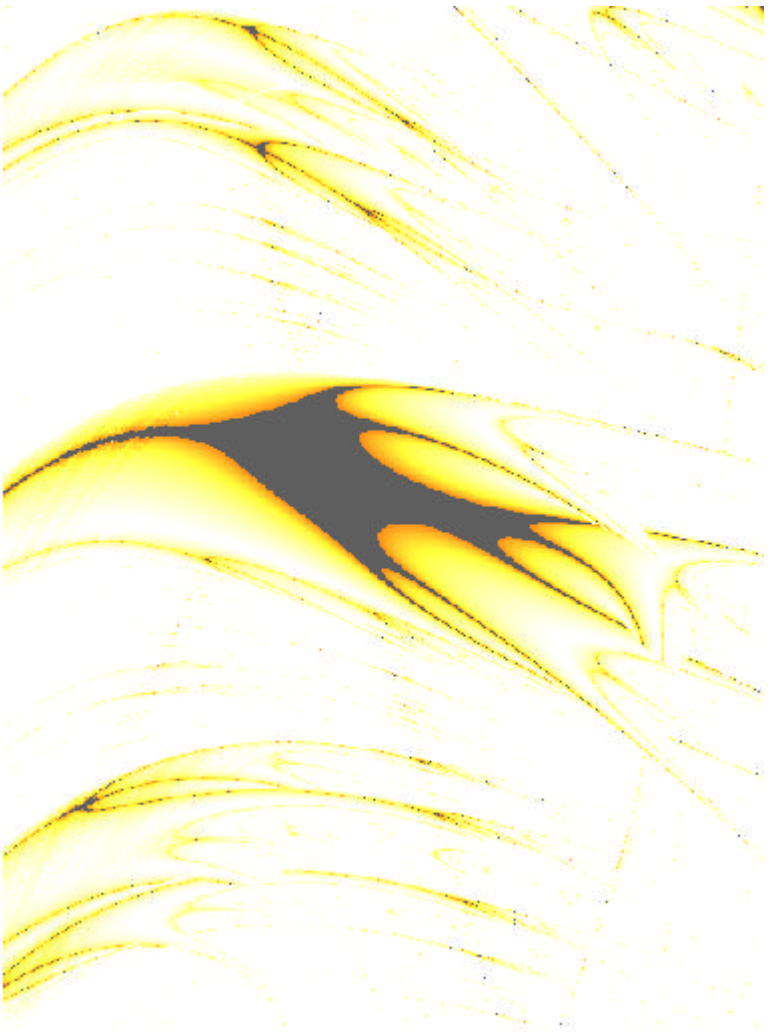
CHAPITRE 11

UML, les diagrammes de composants

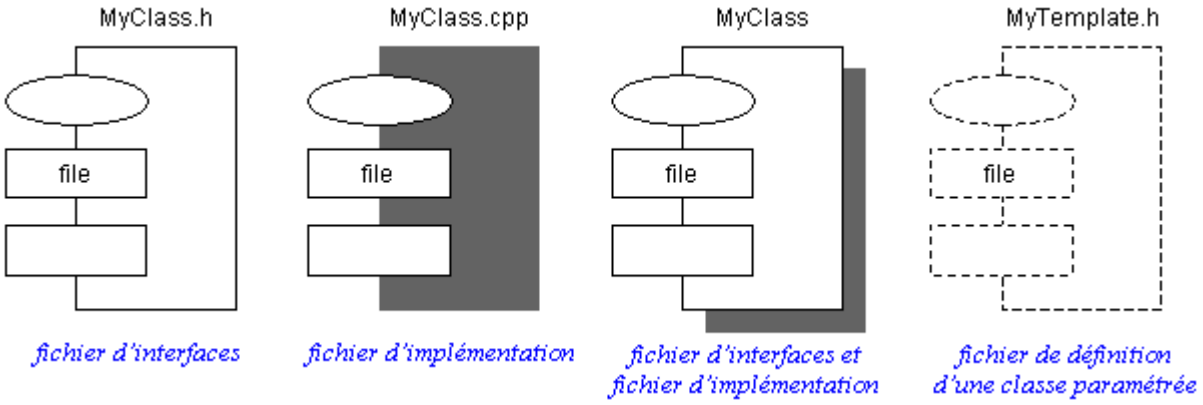


11.1 *But du diagramme de composants*

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules : fichiers sources, librairies, exécutables, etc. Ils montrent la mise en oeuvre physique des modèles de la vue logique avec l'environnement de développement.

Les dépendances entre composants permettent notamment d'identifier les contraintes de compilation et de mettre en évidence la réutilisation de composants. Les composants peuvent être organisés en paquetages, qui définissent des sous-systèmes. Les sous-systèmes organisent la vue des composants (de réalisation) d'un système. Ils permettent de gérer la complexité, par encapsulation des détails d'implémentation.

FIGURE 11.1 Modules (notation)



11.1.1 *Relations entre composants*

Les composants d'un logiciel possèdent des relations entre eux; la majeure partie de ces relations consistent en des contraintes de compilations et d'éditions de lien. A ce type de relation correspond essentiellement un équivalent des fichiers "make" (*Makefile*) de UNIX. La difficulté des fichiers make est de bien documenter les dépendances, de manière à être sûr de toujours recompiler les éléments nécessaires (dépendences pas forcément triviales, au moyen de pointeurs en C++).

Dans le cas de projets de dimension limitée, il est possible d'omettre ce genre de relations de dépendance en recompilant systématiquement l'ensemble du code; le diagramme de composants devient alors de peu d'utilité; néanmoins, ce diagramme permet de visualiser plus facilement qu'au travers d'un fichier make les dépendances, lors de la modification d'un composant par exemple. Il devient ainsi possible de suivre relativement aisé les conséquences d'une modification au ravers des divers composants du code.

Les relations peuvent -et devraient- être documentées (figure11.2, page113).

FIGURE 11.2 Diagramme de composants