Préface

par Luc Hermitte

Le C++ est un paradoxe.

Le C++ est un langage extrêmement riche et complexe. Définitivement complexe. Il faut bien l'admettre. Qui n'a jamais pris peur en découvrant les templates pour la première fois ? Quel développeur venant du procédural ne s'est jamais retrouvé perdu devant des objets ? Vous avez aussi certainement pesté, ou entendu pester, après le C++ (et/ou le C) pour ses plantages inopinés, et cette mémoire qu'il faut encore, au XXI^{e} siècle, gérer à la main.

Outre son lot de complexités intrinsèques, de par les choix faits sur le langage et sa syntaxe, le C++ traîne les difficultés qu'il hérite du C. Ai-je besoin de rappeler les conversions implicites qui aiment nous prendre en traître ? L'attention particulière qu'il faut sans cesse accorder au bas niveau et toutes ces petites étoiles qui parsèment nos codes ? Et ce n'est là que le début des complexités que traîne le C++. Il faut également compter avec la complexité du génie logiciel, que cela soit au niveau algorithmique, comme dans les processus de la conception à la livraison en passant par les tests. De plus, chaque paradigme que le C++ supporte présente ses difficultés. Comment concevoir correctement des hiérarchies de classes qui interagiront entre elles ? Comment écrire des métaprogrammes ? Ou encore comment intégrer la programmation fonctionnelle ? Comment mélanger ces paradigmes, ou devrions-nous dire *styles de programmation*, en un seul qui profite de chacun de façon opportune ? Enfin, dans un projet il y a aussi les difficultés propres au métier. Sans parler des cas dégradés à impérativement prendre en compte quand on travaille sur des systèmes critiques.

Mais tout en étant complexe, le C++ peut aussi être simple à utiliser. Bien plus que le C par exemple, qui pourtant est par nature beaucoup plus simple. Là est le paradoxe. Ce langage qui est tout sauf simple peut être utilisé de manière simple. Néanmoins pour y parvenir, il faut se donner un cadre, des bonnes pratiques, et s'y tenir. J'ai envie de prendre l'exemple d'un collègue qui me disait pas plus tard qu'il y a deux jours : "Depuis que j'utilise le RAII dans mes programmes, je n'ai plus une seule fuite de mémoire." Une technique, simple, et voilà : un des fléaux du C, et du C++ est contenu et maîtrisé.

Ces bonnes pratiques, si on veut en profiter, il faut investir du temps pour les apprendre. Fort heureusement, beaucoup ne sont pas spécifiques au C++. Un développeur professionnel devrait idéalement connaître les bonnes pratiques en rapport avec le génie lo-

giciel, ou avec la programmation orientée objet, voire être initié à la programmation par contrat. Le contexte C++ est presque un détail relativement à ces aspects-là. Il s'agit essentiellement de rajouter quelques idiomes du C++, comme le RAII, et le tour est joué.

Pourquoi apprendre toutes ces bonnes pratiques ? Pour savoir écrire un code robuste, qui soit simple non seulement à écrire, mais aussi et surtout pour qu'il soit simple à maintenir.

D'accord, mais comment les apprendre me demanderez-vous ? Vous devez vous douter de ma réponse, car vous avez ouvert le bon livre. En effet, Philippe Dunksi y traite les bonnes pratiques du C++. Vous n'en sortirez pas avec une liste de points à respecter pour passer le contrôle qualité sur vos projets, mais avec un savoir qui vous permettra de mettre au point des programmes robustes et simples, dans la limite de la complexité inhérente au métier de vos applications. Un savoir critique, car aucun outil (autre que des revues de pairs) ne permettra de détecter les fautes de conception contre lesquelles cet ouvrage va vous apprendre à vous protéger.

Que vous soyez développeur, concepteur ou architecte du secteur logiciel, si vous travaillez en C++, ce livre est pour vous. À vrai dire, les deux premières parties vous concerneront quel que soit le langage que vous utilisez. En particulier, la deuxième partie est pertinente indépendamment du langage orienté objet que vous employez. En effet, pour l'essentiel, ce qui y est présenté transcende les langages. Vous trouverez en ce livre un compagnon précieux qui viendra compléter ce que vous avez déjà appris sur le C++.

Sans plus attendre, je vous laisse découvrir le "C++ moderne".

Toulouse, le 24 janvier 2014