Przedmowa

John Vlissides

Czy można napisać o C++ coś, czego jeszcze nie napisano? Okazuje się, że tak – i to bardzo dużo. Książka jest splotem kilku technik programistycznych – programowania uogólnionego, metaprogramowania szablonowego, programowania obiektowego i stosowania wzorców projektowych. Każda z tych technik z osobna jest już dobrze znana, ale ich połączenie dopiero zaczynamy doceniać. Synergia ta otworzyła przed C++ zupełnie nowe horyzonty nie tylko w dziedzinie tworzenia kodu, ale również projektowania, analizy i architektury systemów informatycznych.

Zaprojektowane przez Andreia komponenty generyczne podnoszą poziom abstrakcji tak, że C++ zaczął przypominać język specyfikacji[1]. W odróżnieniu od wyspecjalizowanych języków specyfikacji, zachował jednak pełną siłę wyrazu. Andrei pokazuje, jak programować w języku koncepcji projektowych: singletonów, odwiedzania (ang. *visitation*), pełnomocników (ang. *proxies*), fabryk abstrakcyjnych (ang. *abstract factories*) i innych. Jak za pomocą parametrów szablonów dowolnie zmieniać szczegóły implementacyjne, i to bez żadnych wręcz kosztów związanych z czasem wykonania. Nie trzeba przy tym wykładać grubej forsy na nowe narzędzia ani przebijać się przez tomy naukowego grochu z kapustą. Potrzebny jest jedynie zaufany, w miarę nowy kompilator C++. No i ta książka.

Generatory kodu przez wiele lat zapowiadały się równie obiecująco, ale na podstawie badań oraz doświadczenia przekonałem się, że w ostatecznym rozrachunku przegrywają. Pojawiają się problemy: którym źródłom odpowiada wygenerowany kod; jest zbyt mało kodu, by pisać generator; generator nie daje się dopasować; wygenerowany kod jest niejasny i wygenerowanego kodu za cholerę nie da się zintegrować z tym, który mamy. Każdy z tych problemów może zniechęcić, wszystkie razem sprawiają, że generowanie kodu jest nieopłacalne w odniesieniu do większości wyzwań programistycznych.

Czyż nie byłoby wspaniale, gdyby urzeczywistniły się teoretyczne korzyści płynące z generowania kodu (szybsze, łatwiejsze programowanie, ograniczona redundacja, mniejsza liczba błędów), ale bez żadnych niedociągnięć? Takie właśnie podejście prezentuje Andrei. Komponenty generyczne za pomocą dających się łatwo składać szablonów implementują struktury programu. Robią to, co generatory kodu: tworzą sztancę niezbędną z punktu widzenia kompilatora. Różnica polega na tym, że robią to w ramach C++, a nie poza nim,

[1] W odróżnieniu od klasycznego zastosowania C++ jako języka implementacji. (Przyp. tłum.)

skutkiem czego integracja z resztą kodu jest bezbolesna. Możliwe jest także wykorzystanie pełnej siły języka do rozszerzania, przesłaniania i rozmaitego dopasowywania komponentów do swoich potrzeb.

Trzeba przyznać, że niektóre z przedstawionych w książce technik są trudne. Dotyczy to zwłaszcza metaprogramowania szablonowego omówionego w rozdziale 3. Opanowanie ich daje jednak bardzo solidne podstawy do tworzenia potężnej konstrukcji z komponentów generycznych, która jakby od niechcenia powstaje w następnych rozdziałach. Śmiem twierdzić, że sam materiał z rozdziału 3 jest wart ceny tej książki, a jest przecież jeszcze dziesięć innych, równie wartościowych. „Dziesięć" to już rząd wielkości. Prawdziwy zysk okazuje się jednak o wiele większy.

John Vlissides
IBM T. J. Watson Research
Wrzesień 2000